2023

Improving the Flexibility and Robustness of Machine Tending Mobile Robots

Richard Ethan Hollingsworth
Virginia Commonwealth University

VCU College of Engineering
Electrical and Computer Engineering

Improving the Flexibility and Robustness of Machine Tending Mobile Robots

Virginia Commonwealth University
College of Engineering
Department of Electrical and Computer Engineering
601 West Main Street
Richmond, Virginia 23284–3068

Improving the Flexibility and Robustness of Machine Tending Mobile Robots

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Science at Virginia Commonwealth University.

by

Richard Ethan Hollingsworth
B.S. Computer Engineering, Virginia Commonwealth University, 2021
A.A.S. Mechatronics, Tidewater Community College, 2015

Director: Patrick Martin, Ph.D.
Assistant Professor, Department of Electrical and Computer Engineering

Virginia Commonwealth University
Richmond, Virginia
May, 2023

Acknowledgement

Many thanks to the numerous people who have helped and encouraged me on this lengthy journey. It began with my parents, and would not have been possible without their unflagging belief and support for all of these years.

Thank you to my advisor Patrick Martin. Your advice, consult, and patience has been invaluable these past few semesters. Fingers crossed that we have the opportunity to work together on additional projects in the future.

And most of all, thank you to my beautiful wife. You are the best thing that ever happened to me and the reason I am here now. Your encouragement, belief and support has kept me going in even the hardest times. You are truly amazing.

Contents

List of Figures

Abstract

Improving the Flexibility and Robustness of Machine Tending Mobile Robots

By: Richard Ethan Hollingsworth

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2023.

Major Director: Patrick Martin, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

While traditional manufacturing production cells consist of a fixed base robot repetitively performing tasks, the Industry 5.0 flexible manufacturing cell (FMC) aims to bring Autonomous Industrial Mobile Manipulators (AIMMs) to the factory floor. Composed of a wheeled base and a robot arm, these collaborative robots (cobots) operate alongside people while autonomously performing tasks at different workstations. AIMMs have been tested in real production systems, but the development of the control algorithms necessary for automating a robot that is a combination of two cobots remains an open challenge before the large scale adoption of this technology occurs in industry. Currently popular docking based methods require a mount point for the docking station and considerable time for the robot to locate and park. These limitations necessitate the consideration and implementation of more modern robot control and path planning techniques. This work proposes and implements a simulation testbed that uses a contemporary whole-body control, OCS2, to perform more flexible pick-and-place tasks. Within this testbed, an Industry 5.0 based pick-and-place framework is deployed, fine-tuned and tested. This system supports the one-shot lead-through based assignment of a prepick position by an operator, thus enabling the cobot to drive to this position and successfully pick up the part agnostic of base orientation and/or position. The proposed system allows robot path planning experimentation and assessment against a variety of cost and constraint values, and is capable of being modified to support various vision based part locating algorithms.

Chapter 1

Introduction and Background

1.1 Manufacturing in the Digital Age

Since its advent in 2011, the term Industry 4.0 has become synonymous with any manufacturer attempting to attain machine intelligence and connectivity. Industry 4.0 includes smart manufacturing, smart factories, cyber physical systems, artificial intelligence, IoT/IIoT, and cloud computing [1], [2]. The realized benefits for companies that have adopted 4.0 approaches includes improved process line efficiency, flexibility and agility, improved product quality, and increased profitability through reduced costs and higher revenues [3]. The 4.0 revolution brought the internet and networking to the factory floor. The "digital thread", an industry term referring to "a communication framework that connects traditionally siloed elements in manufacturing processes and provides an integrated view of an asset throughout the manufacturing lifecycle" [4], enabled machine to machine communication and awareness.

Though many companies remain in the process of adopting 4.0 standards, its proponents and early adopters have come to realize that a crucial piece was missing; the human. Or more specifically, a focus and consideration on how machines and humans interact intelligently and cooperatively. Because although machines are great when everything is going as planned, when crisis hits the production line, many machines lack the flexibility to diagnose and

correct the problem.

Industry 5.0 keeps what works [5] (the focus on intelligent machines, etc.) and improves on what does not (the perceived discarding of the human worker) [6]. 5.0 is a human-centric design solution where humans and collaborative robots (cobots) work hand in hand on production tasks. The cobot will handle repetitive tasks and labor-intensive work, while the human takes care of customization and critical thinking [6], [7]. Combining human expert creativity with efficient, intelligent, and accurate machines, 5.0 aims to produce more resource-efficient and user preferred manufacturing solutions as compared to 4.0 [7].

This amalgamation of human and cobot is encouraging research and innovation in what is termed the flexible manufacturing cell (FMC) [8]. Initially designed around 4.0 based digital technology, these cells were envisioned to comprise a variety of machines capable of peer-to-peer communication, learning, and adaptation with the end goal of creating an autonomously managed work cell free of any human influence. For example, an autonomous machining cell comprised of a CNC machine for cutting, a CMM machine for measuring, and a mobile manipulator cobot for inbound/outbound part movement, could be independent and capable of mass part production. The digital thread based 4.0 technologies enable much greater process management and control, but fail to completely account for potential failures, changeovers, unencountered situations, or customer required human oversight. Thus, Industry 5.0 with its focus on human-machine collaboration is required to fully address the needs of a truly flexible manufacturing cell [9].

1.2 Autonomous Industrial Mobile Manipulators

The 5.0 approach to human-robot collaboration brings the individual into the workcell with the cobot. Traditionally, these workcells consist of a fixed base robot repetitively performing a task it had been programmed for by a human. There would usually be a safety envelope surrounding the cell to ensure the human was not placed in danger from a robot

unaware of their presence. The robot would become part of the workcell, inflexible and fixed, performing its function until the production line was shut down, the workcell removed, and the robot along with it.

As the manufacturing industry transitions from mass production (i.e., repeatedly producing the same good for years) to mass customization (i.e., producing a customer specific part), the flexible manufacturing cell will turn the rigidity of the typical production line on its head. But for a cell to truly be "flexible", the fixed base manipulators must be replaced with something more adaptable. Therefore, Autonomous Industrial Mobile Manipulators (AIMMs) are being introduced to the factory floor. AIMMs bring greater flexibility, variety, and collaboration to the manufacturing industry. Composed of a wheeled base and a robot arm, these cobots operate alongside people while autonomously performing tasks at different workstations [10], [11]. Capable of mapping and intelligently moving around an industrial environment through its localization [12], these robots offer all the strengths of traditional fixed base manipulators (e.g., the ability to repeatedly perform simple tasks), while removing many of the weaknesses (e.g., the inability to operate across processes and locations).

AIMMs have been tested in real production systems, but there are many open challenges before the large-scale adoption of this technology occurs [10] in industry. One of the greatest challenges involves the development of the control algorithms necessary for automating a robot that is a combination of two cobots; a mobile vehicle and a fixed base arm. Many repetitive production tasks assigned to robots in industry involve what is known as "pick-and-place" operations. This task simply picks up an object from one location and places it in another location. While a repetitive task for a human, it is one that robots excel at and for which an AIMM, with the removal of its fixed base restriction, can further enhance through the management of multiple workcells and/or the movement of parts between distant locations.

For a fixed base cobot arm, a pick-and-place operation is easily accomplished as the robot always knows where it is in space, and operator taught pick and prepick positions are

easily repeatable using well known inverse kinematic based control algorithms [13]. However, the introduction of a mobile base into the pick-and-place operation complicates things as the robot arm has increased uncertainty of where its base is located in space, thus rendering obsolete many of the traditional lead-through [14] based programming techniques and inverse kinematic based control solutions. Additionally, the 4.0 and 5.0 revolutions mandate that all machines possess a level of intelligence which enables them to behave autonomously and safely amongst humans. Much research exists on how to drive somewhere with a mobile robot and there is a lot of research on how to intelligently pick up a part with a fixed base robot. However, there is minimal work that concerns how a mobile manipulator could reliably and repeatedly pick up that part in a safe and efficient manner among human partners.

1.3 Anatomy of a Mobile Pick-and-Place

The picking of an object by a mobile manipulator is a problem composed of two simpler tasks/problems with well known answers.

1. Q: How do you drive to the part? A: There are many well known mobile path planning algorithms [15].
2. Q: How do you pick up the part? A: Treat the robot as a fixed base manipulator once it is within a reachable distance of the part.

Typical solutions to these problems utilize work station mounted docking ports. The mobile base of the robot can autonomously guide itself into these docks to provide consistent docking for the AIMM. Issues with this approach include nonrobust and/or slow path planning for the mobile drive, along with much time being required for the autonomous docking. After docking, the robot is treated as a fixed base and a previously taught point is used to pick up the part. However, even with lidar based docking stations, and ignoring the slowness of the dock, there remains inaccuracy involved for repeatable docking. Thus, docking port pins [16] are used for more precise docking. This solution only leads to even

more slowness as the robot must precisely line itself up with the pins before it is able to park. What is needed is a framework that provides a solution set for both problems 1 and 2.

1.3.1 Part Navigation

Problem 1 describes a "gross mobile path planning" problem which should be solved by an algorithm capable of *quickly and robustly* moving the robot within arm's reach of the part. Robustly will be defined as the ability to safely drive to the part and park without concern for exact final positioning of the mobile base. This robustness will deliver the robot to the pick position quickly if both arm and base motions are performed smoothly, without collision, and without final park position or orientation constraints, other than that the arm is within reach of the part.

Fixed base collaborative robot arms are typically trained with lead-through [14] based programming, a technique where the robot joints are unlocked and a robot operator physically moves the robotic manipulator through the waypoints of a desired task. Traditional pick-and-place operations in industrial environments usually involve the teaching of an end-effector prepick position above the part that the robot can move to before performing the actual pick. The fixed base nature of traditional cobot arms makes this an effective and efficient robot training approach. Additionally, in the production industry, there is still much concern and suspicion about "fully autonomous" machines being outside of human definition and control. Thus, the lead-through teaching of a cobot aligns neatly with the Industry 5.0 goals of letting the human be the brains and the robot be the brawn.

With the goal of keeping the human in the loop of the robot training, any mobile path planning pick-and-place algorithms utilized should allow the assignment of a point in space, relative to the end-effector, to which the robot should move. There are many individual path planning solutions for a mobile base (RRT, A*, etc.) [17] which do not consider a mobile manipulator with an arm. There are also many well-documented arm based path planning solutions which do not consider a mobile base [18]. Because of the requirement that the arm

be in a position taught by the operator from which the part is picked up, the result of these individual mobile path planning algorithms is that the gross path planning would then have to be further deconstructed into two control techniques, one for the base and one for the arm. Two teach points would then be required for this setup, one for moving the base to a specific park position with a given orientation and position and another for moving the arm to the prepick position which is a recorded taught point as well. This setup is feasible, but the programming complexity and overall flexibility of the system is reduced.

A better system would allow the assignment of a single point in space, relative to the end-effector, to which the robot can move. As a counterpoint to programming a mobile manipulator as two separate entities and teaching two different points, whole-body manipulation techniques are now being explored. These methods allow for the assignment of a single target and treat the mobile base and arm as one system which trajectories are computed for and movement commands sent to [19], [20]. Such a whole-body system which implements trajectory planning via model predictive control (MPC) is OCS2.

MPC is a control method that uses a system model to predict future behavior by solving online a constrained, discrete-time, optimal control problem. The solution returns a finite sequence of control actions from which the first control is applied at state x [21], [22]. MPC controllers are well suited for robotics applications due to their ability to handle constraints and run in real-time.

OCS2 is an MPC based software framework which provides tools to set up the system dynamic models and cost/constraint functions from a URDF model of the robot.[1] These constraint functions are built into OCS2 for end-effector tracking and control of a mobile manipulator. Using this framework in an Industry 5.0 production environment would allow the operator to use lead-through programming for the assignment of a single point in space relative to the robot's end-effector. As long as the AIMM is localized in a world map and has a point reference for where the end-effector is in space (x, y, z, quaternion coordinates), it

[1] OCS2 repository

can compute velocity commands for both base and arm simultaneously when returning the end-effector to that assigned point without the constraint of base positioning or orientation having to be repeatable. This allows for a faster and more flexible positioning system that does not rely on the precise replication of the base's original position when the end-effector goal point was assigned. Computed paths to the goal position are smoother and swifter due to the removal of this constraint.

1.3.2 Part Picking

With the assumption that the mobile base has driven the end-effector to the prepick position, and within a Euclidean based reachable distance of the part, an algorithm capable of reliably picking up the part from a variety of initial base positions and orientations can be utilized. This algorithm should be quick to train and easy to deploy in production by nonexpert personnel. The indeterminate nature of the base position when parking requires that the selected picking algorithm possess some degree of intelligence and environmental awareness.

Reinforcement learning techniques are being explored for the adding of this intelligence to the robot. SAC [23], PPO [24] and world models [25] are three of the most popular RL methods for enabling a robot to learn how to pick up an object. The downside of these and other RL algorithms is that they typically require large datasets and/or training time as the entire state space of the arm needs to be discovered and explored before the arm learns to pick up a part. Additionally, much of the most effective and cited work within this space has been performed with fixed base manipulators and an unconstrained state space (i.e., the robot can explore anywhere within its reach when learning to perform a task).

GUAPO [26] is a technique descended from RRL [27] which aims to address the slowness of RL by defining an "uncertainty region" around the part and only learning a policy for this region, while using typical model based policies for the arm when moving outside of that region. In demonstrations, GUAPO learned faster than SAC, but still required at least a

half hour of training for a simple peg insertion task. Additionally, the DOPE [28] algorithm required for identifying the part location and defining the uncertainty region required eight hours of offline training, while the VAE [29] network used on the eye in hand camera required an additional 160,000 datapoints and twelve epochs of offline training. Thus, while GUAPO can theoretically learn to perform a peg insertion task in thirty minutes, this does not account for the offline time required to train the camera networks for identification of said part.

The creators of region limited residual reinforcement learning (RRRL) [30], perhaps recognizing the inefficiency of training the camera networks, used a pure learning from demonstration (LfD) [31] based movement policy for the gross motion of getting to the part. They then combined this with a force based learned policy for the residual part. While faster to train than GUAPO due to the elimination of the camera networks, the learned policy used a suction cup based end-effector for picking up a piece of cardboard (a use case optimized for this type of policy). Unfortunately, a force feedback policy is not well suited to learning the positioning required for the picking up of an object.

The requirement that the robotic pick be unaffected by initial mobile base orientation and position leads to the conclusion that the algorithm must contain some sort of intelligent visual feedback. However, traditional RL algorithms and residual learning algorithms still require many hours to train. In industry, asking operators to sit back and wait for the robot to learn how to pick up a part or perform a task is something that would be looked on with suspicion and/or outright refused if suggested. Therefore, any solution for the part picking deployed in industry needs to be fast, reliable, and repeatable. Possible options include:

- **Sim to real transfer** – Robotics simulators such as Gazebo and Isaac-Sim are increasingly focusing their efforts on simulated data production and training with the goal of ultimately being able to fully train in simulation before transitioning to reality. This goal requires the development of an environment which replicates the "real-world" for the robot to train in [32], [33]. While physics based simulators have improved dramatically, the effort required to create a model that is reflective of the actual workcell

is still quite intense.

- **Visual servoing** – Provides robot control based on visual feedback from RGB or RGBD images [32], [34]. While the segmentation and identification algorithms are slow and compute heavy, past work has demonstrated the ability to detect and pick a known part.

- **Robot Grasp Pose Detection (GPD)** – Utilizes a pretrained CNN to produce a number of potentially viable grasps from a single point cloud image [32], [35]. Downside is that if the object is not segmented properly from the background, the technique could be compute heavy and potentially slow to provide a viable grasp if there is not enough processing power available on the system.

1.4 Research Objectives

This research investigates a more robust pick-and-place framework for a machine tending mobile robot. Specifically, the picking behavior of a mobile robot will be investigated here. Rather than using inefficient docking ports or reinforcement learning techniques requiring lengthy training times, the mobile pick operation will be decomposed into the two subtasks as discussed in Section 1.3.

The mobile part of the pick will utilize the OCS2 whole-body manipulation framework. Once the cobot has driven the end-effector to the assigned prepick position, the identification and picking of the part will be accomplished with the previously discussed grasp pose detection framework. All experiments will be performed in the Isaac Sim robotics simulator.[2] A custom cobot comprised of a Kinova Gen3 7dof robot arm and a Clearpath Husky mobile base will be created and deployed in the simulator for testing and evaluation.

This research will attempt to answer the following questions:

[2] Isaac-Sim

1. Can this system support the one-shot lead-through based assignment of a prepick position by an operator, thus enabling the cobot to drive to this position and successfully pick up the part agnostic of base orientation and/or position?
2. Can this system perform the task faster (defined as amount of time taken to undock $\rightarrow$ drive $\rightarrow$ dock $\rightarrow$ pick) than docking based methods?

The answers to these questions would be useful in an Industry 5.0 compliant flexible manufacturing cell such as the one being investigated by the Commonwealth Center for Advanced Manufacturing.[3] Traditional pick-and-place operations in industry machine cells have involved the use of fixed base manipulators capable of only performing a single task. This work has the potential to further mobile manipulator ease of training and performance efficiency in distributed manufacturing cells.

The remainder of this work is structured as follows: Chapter 2 covers the creation of a mobile manipulator and the development of a simulated testbed within which a whole-body OCS2 based pick-and-place framework is deployed, tested, and evaluated. Chapter 3 contains a quantitative evaluation of the developed framework, along with analysis of the robot's performance under a variety of cost and constraint values. Chapter 4 is the conclusion and offers a perspective on possible future research paths stemming from this work.

[3] CCAM flexible manufacturing cell

Chapter 2

Whole-body Based Pick-and-Place Framework

2.1 Overview

For autonomous industrial mobile manipulators to be adopted and accepted in the manufacturing industry, it is required that they be flexible, reliable, robust, and safe. The traditional fixed base manipulators check all of these boxes, but the common deployment of a docking based mobile manipulator does not. While reliable and safe, the docking station used with most AIMM deployments limits both the flexibility and robustness of the mobile manipulator solution. As seen in Figure 2.1, these docking stations require a not always dynamically feasible mount point. Additionally, the lidar based docking station has a positional tolerance of approximately ± 5 mm, and both docking stations require considerable time to locate and dock.

Addressing these limitations necessitates the consideration and implementation of more modern robot control and path planning techniques. This work proposes and implements a simulation testbed that uses a contemporary whole-body control to perform more flexible pick-and-place tasks. Within this testbed, the Industry 5.0 based pick-and-place framework

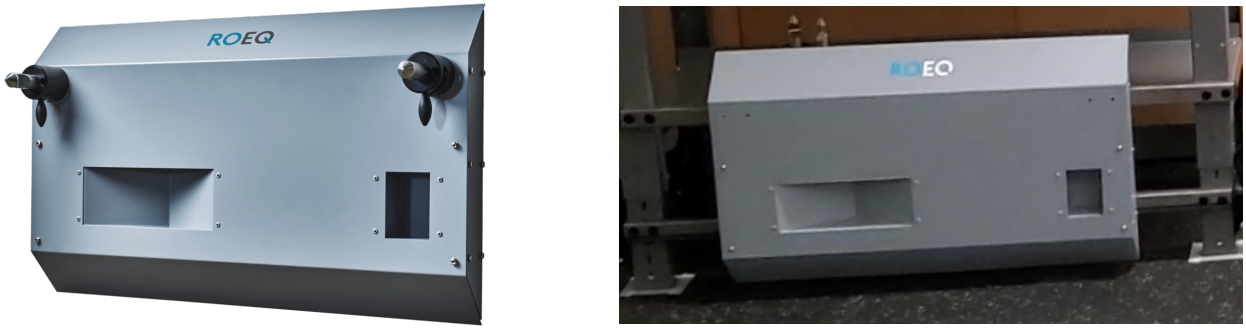


Figure 2.1: Examples of a precision docking station (left) and mounted lidar based docking station (right).

is deployed, fine-tuned and tested.

The integration of numerous softwares, technologies, and languages is required for the creation of any useful framework and testbed. Figure 2.2 offers a visual overview of the technologies and control stack used for creating a human assisted whole-body control based pick-and-place architecture. As discussed in Section 1.3, a mobile pick-and-place is decomposed into a part navigation task and a part picking task. The part navigation task requires a control framework which is robust, reliable, and docking station independent. For this work, the OCS2 [36] model predictive control whole-body based framework was selected. The part picking task requires a system capable of identifying and picking a part from a variety of positions, and for this work a custom OpenCV based vision node was written and deployed. For the testing and evaluation of the pick-and-place framework, a custom built testbed designed within Nvidia's Isaac-Sim robotics simulator was designed and leveraged. Each one of these pieces will be discussed in the following sections.

2.2 Robot Simulation

The Robot Simulation node is the heart of the pick-and-place testbed, and responsible for the utilization of both the MPC and vision nodes. Explored, developed against, and leveraged in this work was Nvidia's Isaac-Sim, a robotics simulator which provides features for building physically realistic virtual robotic worlds and experiments. Isaac supports ROS/ROS2 based

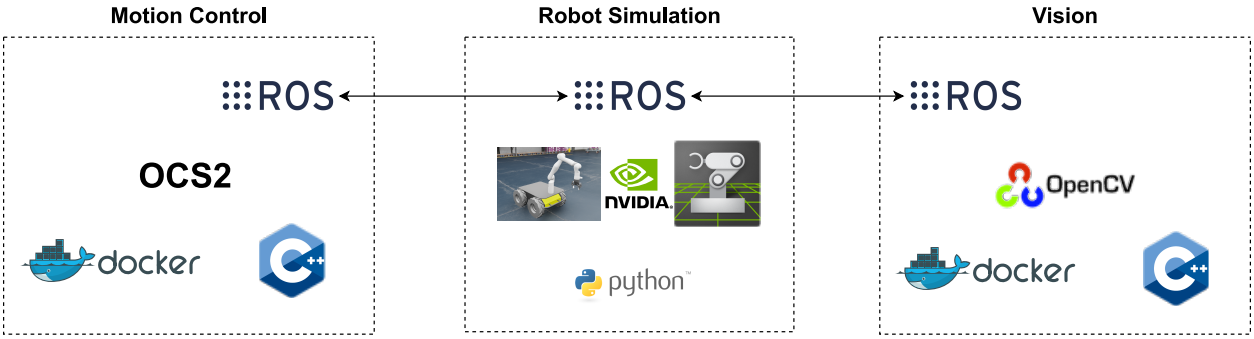


Figure 2.2: System components of the pick-and-place testbed.

navigation and manipulation applications, and simulates sensor data from sensors such as RGB-D, Lidar and IMU.

Leveraging a simulator such as Isaac-Sim enables the evaluation of frameworks and algorithms in an environment that may not otherwise be accessible. This thesis creates and evaluates an Industry 5.0 based pick-and-place application. As such, building a physically accurate environment which allows the rapid altering and testing of said framework is quite valuable.

The Isaac-Sim deployment in this work leverages ROS Noetic for communication with the simulated robot's control nodes. A Python API is provided by Nvidia which enables researchers to prototype and deploy custom built resources, environments and algorithms. All resources and scenes within Isaac-Sim are represented in the USD interchange file format. Universal Scene Description (USD) is a Pixar created open-source 3D scene description and file format used for content creation and interchange between different tools. The custom creation and deployment of multiple USD assets to a scene enables the building of a simulated environment that runs on the PhysX physics engine for dynamically accurate environments.

2.2.1 Robotic System

Building the mobile manipulator is the first task in creating a physically accurate testbed for evaluation of any picking framework. AIMM research focuses on the development of integrated robotic systems capable of performing work and/or assistance in manufacturing

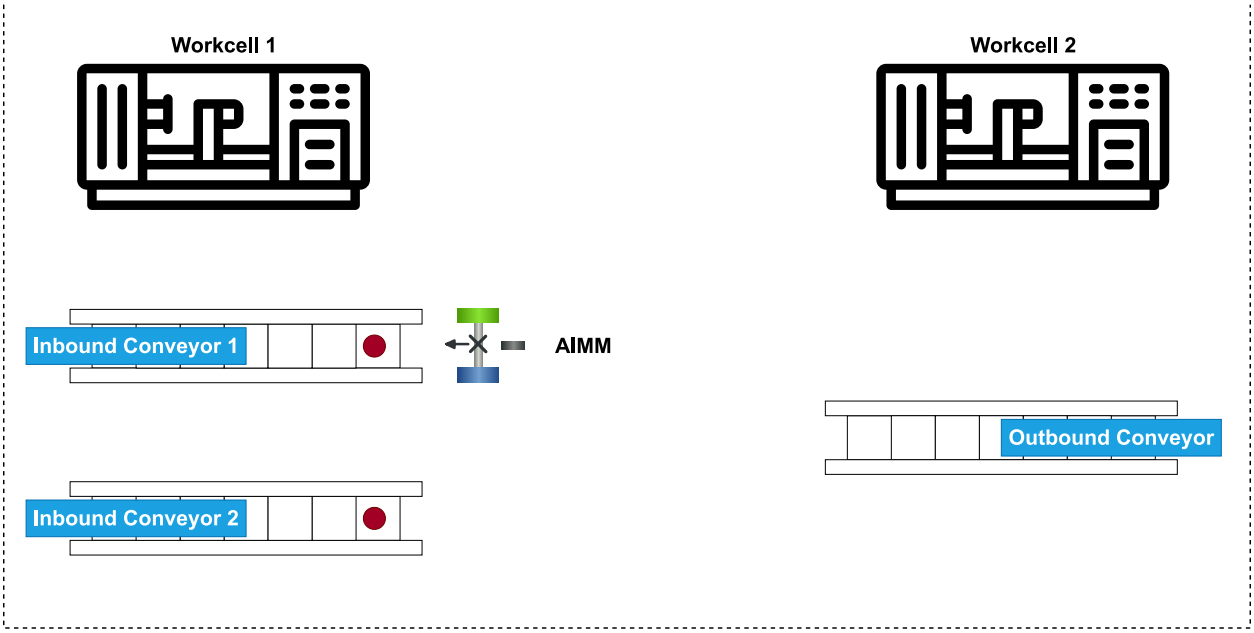


Figure 2.3: Typical example of a pick-and-place based machine tending AIMM.

environments [37]. Picking objects from different workstations, followed by placing said objects at a centralized outbound location, is a typical manufacturing task where an AIMM would be expected to excel. Figure 2.3 illustrates a flexible manufacturing cell in which a robust, nondocking station based mobile manipulator is useful. With multiple inbound conveyors and workstations separated at a greater distance than a fixed base manipulator could handle, the AIMM helps improve the efficiency, cost, and speed of the production cell.

The Clearpath Husky is a four wheel mobile base platform with a differential drive steering system. All four wheels are fixed, which classifies the vehicle as skid-steer due to the fact that turning is done by varying the different wheel speeds, thus causing the robot to "skid" as it turns. The base of the Husky is wide and stable, providing opportunities for integration with many types of manipulator arms. Additionally, Clearpath provides built-in support for ROS communication stacks, thus simplifying integration with other ROS based systems.

The KinovaGen3 7-DoF manipulator arm is a seven degrees of freedom, lightweight robot arm capable of being mounted onto a Husky base. Like Clearpath, Kinova provides a ROS

Figure 2.4: Clearpath Husky base combined with a KinovaGen3 7DoF arm to form a Husky-KinovaGen3 7DoF Mobile Manipulator.

package which enables communication and control of the robot arm. The arm features an embedded 2D/3D vision capable eye-in-hand camera. This camera sits on the end-effector and is placed to enable vision based pick-and-place tasks.

Figure 2.4 combines the Husky base and the KinovaGen3 arm into a single Autonomous Industrial Mobile Manipulator. With no real-world testbed for advanced manufacturing on hand, simulation is used in this work, as it allows the development, examination and testing of an unknown framework in a safe and convenient environment.

The construction of this novel AIMM in simulation requires formal descriptions of its physical properties. The Unified Robotics Description Format (URDF) is an XML specification used to model multibody systems such as robotic manipulator arms. URDF is popular with ROS, as it is a physical description of the robot's joints, motors, mass, etc. Combined with the referencing of CAD/CAM files representing the robot's rigid bodies, the URDF provides the human and simulator critical information needed to understand the physical shape and size of a robot and the robot's capabilities.

Both Kinova and Clearpath provide URDF models of their respective robots. Each URDF file defines the physical and kinematic properties of the robot. The links of the robot are defined as rigid bodies and these links are connected by either prismatic or revolute joints. Beginning from a base frame/body, the URDF proceeds down the kinematic chain of the robot. For example, the Kinova arm begins from the mounted base frame and ends at the

end-effector tool frame.

It was desired that the arm be mounted on top of the Husky base. From the two separate URDF files, a master URDF was created which utilized a "dummy" joint for joining the arm to the base. Isaac-Sim provides URDF to USD import utilities, which were used to import the Husky-KinovaGen3 URDF and turn it into a USD capable of being opened and tweaked within the Isaac-Sim simulator. The combined model loaded within Isaac-Sim is what is seen on the right in Figure 2.4.

2.3 Motion Control

Robotics motion planning is the "process of breaking down a desired movement task into discrete motions that satisfy movement constraints and optimize some aspect of the movement" [38]. Movement constraints are restrictions that narrow down the number of achievable joint motions. These constraints could be mechanical constraints such as joint limits. In optimization problems, these constraints can also be user defined. Examples of such constraints include requirements that the end-effector be moved to a desired position, that certain joint velocity limits are not exceeded, or that the linear/angular velocities of the mobile base do not cross some defined threshold.

Model predictive control's central idea is that a dynamic or kinematic model of the system is used to predict the future evolution of the state trajectories in order to optimize the control signal and account for possible state violations. This is done while bounding the input, through either soft or hard constraints, to the admissible set of values. MPC relies on receding horizon control which derives the optimal sequence of N steps, but only applies the first optimal control movement action. At time step t+1, a new reference signal (state measurements) is received and the optimization is repeated based on the new current states.

Key MPC design parameters include the sample time, prediction horizon, and control horizon. The sample time is the rate at which the controller executes the control algorithm.

A sample time that is robust to disturbances, but does not cause an excessive computation load is desired. In an online MPC solver, the sample time will be equal to the rate at which the reference signal is received from the system. The prediction horizon is defined as the number of predicted future time steps. This is a measure of how far the controller predicts into the future. If this is too short then the controller may drive the system towards something it can not deviate from. Too long, and the computation cost may be too expensive to calculate in real-time. The control horizon is the number of control moves to time step m which are computed by the solver. It is recommended that the control horizon be 10 to 20% of the prediction horizon and have a minimum of 2-3 steps. As only the first two or three control moves have a significant effect on the predicted output behavior, this provides a good trade-off between accuracy and computational cost.

2.3.1 OCS2 Implementation

OCS2 is an MPC based C++11 toolbox which provides an efficient implementation of the Differential Dynamic Programming (DDP) algorithm in continuous-time (known as SLQ) [36], [39]. Sequential Linear Quadratic Model Predictive Control (SLQ-MPC) is a flavor of nonlinear MPC which "performs forward roll-outs of the current control policy over a prediction horizon with the full nonlinear system dynamics" [40]. The linear-quadratic (LQ) approximation of the dynamics and cost are combined with the linear approximation of the constraints and used to update the LQ model. Ricatti-like equations are then solved using the constrained LQ model [24], [41].

The Husky-KinovaGen3 arm has seven degrees of freedom and the base has a nonholonomic constraint as it can turn in place but can not drive sideways. By parsing the Husky-KinovaGen3 URDF within OCS2 on startup, a kinematic model of the robot is loaded into the MPC controller. OCS2 provides a kinematics interface for any named frame in the URDF model based on the Pinocchio[1] library. This interface provides a first-order model of

[1] pinocchio.com

position, orientation error, and velocity for a list of named frames.

The state of the robot arm is defined by its seven joint positions, $q_{arm} = [q_1, ..., q_7]$. The robot base is modeled as a simulated 2D bicycle model. The base state is defined by the base pose and quaternion encoded yaw, $q_{base} = [x, y, \omega]$, in a static world frame. The full robot state is represented as $q = [q_{base}, q_{arm}]^T$. For the seven DoF Kinova arm, this translates to a state vector of length ten. It is this state vector which is the reference signal fed back to the OCS2 MPC solver each time a new optimized trajectory needs to be computed.

The robot is velocity controlled with the desired base input u_{base} represented by the robot's forward linear velocity and angular z velocity such that $u_{base} = [v, \dot{\omega}]$. The desired arm inputs are represented by the seven DoF velocities, $u_{arm} = [u_1, ..., u_7]$. The optimal reference input computed by OCS2 for the Husky-KinovaGen3 is represented as $u = [u_{base}, u_{arm}]$, a vector of length nine. This optimal input vector is returned by the solver after minimization of the constraint bounded cost function.

The general form of the optimal control problem for the end-effector tracking of a mobile manipulator is formulated as:

$$
\begin{aligned}
\min_{u(.)} \quad & \Phi(q(t_I)) + \int_{t_0}^{t_I} l(q(t), u(t), t)\mathrm{d}t & & \\
\text{s.t.} \quad & q(t_0) = q_0, & & \text{Initial state} \\
& \dot{q}(t) = f(q(t), u(t), t), & & \text{System dynamics} \\
& g_1(q(t), u(t), t) = 0, & & \text{State-input equality constraints} \\
& g_2(q(t), t) = 0, & & \text{State-only equality constraints} \\
& h(q(t), u(t), t) \geq 0, & & \text{Inequality constraints}
\end{aligned}
\tag{2.1}
$$

This is a continuous time control problem with the goal of optimizing the finite-horizon optimal control instantaneous cost (or running cost) functional $l(q(t), u(t), t)$ across the control horizon spanning from t_0 to t_I. $\Phi(q(t_I))$ is a nonnegative terminal/final cost that penalizes the state attained at the final time t_I. The objective function l is a sum of squared error

penalized cost terms subject to the state $q(t)$ and input $u(t)$ at time t. The optimal control problem seeks to minimize the cost functional by changing $u(.)$ when attempting minimization. This optimization is done with respect to the state-input equality $g_1(q(t), u(t), t) = 0$, state-only equality $g_2(q(t), t) = 0$, and inequality $h(q(t), u(t), t) \geq 0$ constraints placed on the system. The optimal control solution must also begin from the initial state of the robot as represented by the constraint $q(t_0) = q_0$, and the robot's next state must be a function of the current state, input and time as seen in the system dynamics equation $\dot{q}(t) = f(q(t), u(t), t)$.

The objectives of the Husky-KinovaGen3 kinematic control problem are end-effector tracking, self-collision avoidance, and collision-free navigation. Any target assigned to the mobile manipulator is relative to the robot's end-effector. Subject to both mechanical and user defined constraints, the MPC solver attempts to minimize the distance from the robot's current end-effector pose to the desired one. Deviations of the robot's end-effector pose (position and orientation) from the desired one are penalized.

In addition to the end-effector constraint, the generated motion plans must also respect other constraints. For handling the constraints in OCS2, you can either use hard or soft constraints. No hard constraints were defined in this work. Soft constraints were defined for the upper and lower bounds of the robot arm's joint positions. Soft constraints were also used for bounding the arm DoF velocities and base forward and angular velocities, as well as creating self-collision bounds.

The soft constraint handling is based on the penalty method where the constraints are wrapped with user-defined penalty functions. Penalty methods are used for turning constrained optimization problems into a series of unconstrained ones. A quadratic penalty function was used on all constraints. If the constraint is satisfied, no penalty is added, else a penalty equal to the square of the violation is taken. The violation is also multiplied by a penalty coefficient σ which provides weighting on the importance of the violation. The constraint term and penalty function are combined into a cost term which OCS2 can collect to form a sum of costs optimization problem to be minimized. The general form of an

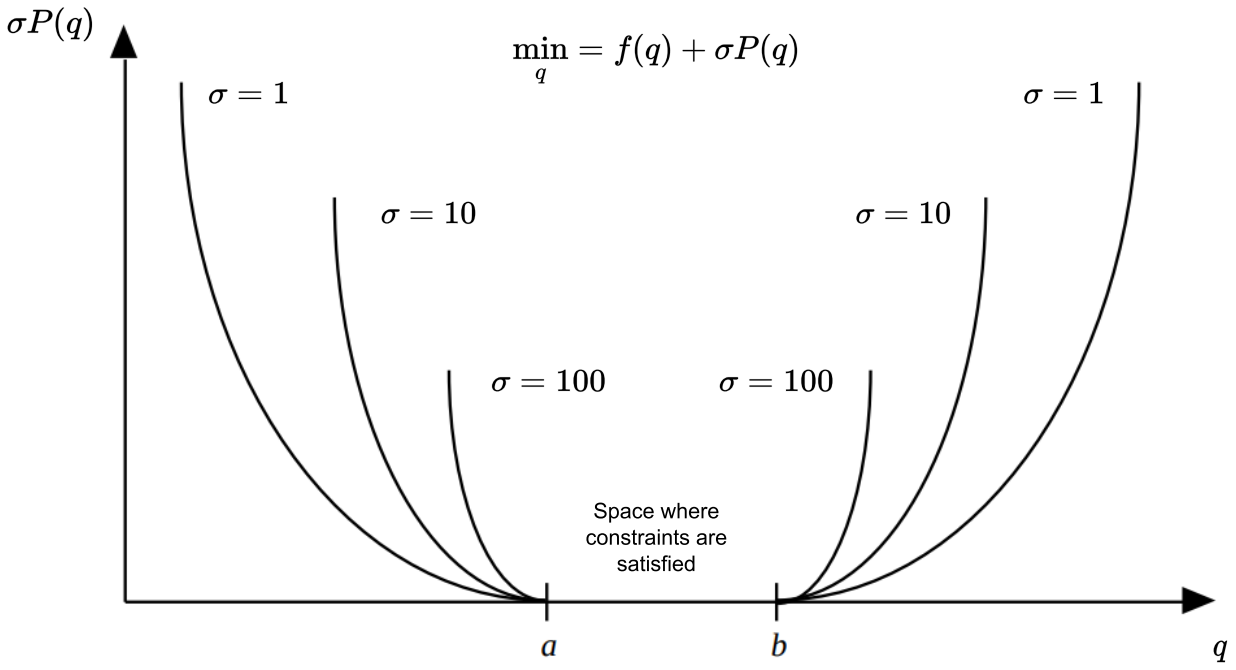


Figure 2.5: Quadratic penalty function effect as sigma increases.

unconstrained minimization problem solved through penalty methods is:

$$\begin{aligned} \min_{\Phi_k(q)} &= f(q) + \sigma_k \sum_{i \in I} g(c_i(q)) \\ g(c_i(q)) &= max(0, c_i(q))^2 \end{aligned} \tag{2.2}$$

In Equation 2.2 $g(c_i(q))$ is the quadratic penalized constraint function and σ_k are the penalty coefficients. In each k iteration the penalty coefficient σ_k is increased by some factor, the unconstrained problem is solved and the solution is used as the initial guess for the next iteration.

A graphical illustration of the effect of an increasing penalty coefficient on a one dimensional problem is shown in Figure 2.5. It is seen in this graph that at increasing values of σ, the corresponding solution will approach the space where the constraints are satisfied and thus minimize f.

The major disadvantage of penalty methods is that one must start near zero and take $\sigma \to \infty$. This is because, although it seems like the problem could be quickly solved by

starting off at a "large" σ, large depends on the model and can not be known beforehand. Additionally, the problem dynamically changes with the relative position of q (i.e., the robot's joint positions are constantly moving) and thus the subset of the constraints that are violated changes. Another difficulty is that at very large values of σ, there are very steep valleys present that present convergence difficulties for all preferred optimization search algorithms. Therefore, σ must begin quite small and gradually increase. This can lead to lengthy solution convergence times and high computational costs.

Thus, OCS2 leverages the Augmented Lagrangian [42] method. This technique is similar to the penalty method, but in addition to the penalty coefficient, an additional term known as the Lagrange multiplier λ is added into the objective function. The Augmented Lagrangian method takes the form:

$$\min_{\Phi_k(q)} = f(q) + \frac{\sigma_k}{2} \sum_{i \in I} g(c_i(q)) + \sum_{i \in I} \lambda_i g(c_i(q))^{\frac{1}{2}} \tag{2.3}$$

$g(c_i(q))$ is defined as in Equation 2.2. After each iteration, in addition to updating σ_k, the Lagrange multiplier λ is updated according to the rule $\lambda_i \leftarrow \lambda_i + \sigma_k c_i(q_k)$. Due to the presence of the Lagrange multiplier, σ can stay much smaller. This helps avoid ill-fitting, and contributes to essentially exact satisfaction of constraints, as well as faster solution convergence as it is no longer necessary to take $\sigma \to \infty$.

In addition to the sum of penalty weighted constraint functions solved by the Lagrangian method within OCS2, there is an additional user defined cost on the control input which is added into the same objective function minimization sum. This cost function takes the form:

$$L = 0.5u'Ru \tag{2.4}$$

The user defined input-cost weighted matrix R is of the form:

$$R = \begin{bmatrix} r_1 & 0 & 0 & \dots & 0 \\ 0 & r_2 & 0 & \dots & 0 \\ 0 & 0 & \ddots & \dots & 0 \\ \vdots & \dots & \dots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & r_9 \end{bmatrix} \tag{2.5}$$

Making the matrix coefficients for certain inputs of R larger penalizes higher DoF velocities for those inputs more severely. This encourages the minimizer of the penalty function to find optimal input solutions which more heavily utilize the DoFs with low input weights. Section 3.3 provides in-depth analysis on the tuning of the cost matrix R defined in Equation 2.5.

2.3.2 OCS2 Integration with Isaac-Sim

Figure 2.6 offers an in-depth view of the Husky-KinovaGen3's Isaac-OCS2-ROS based implementation. The MPC solver of OCS2 is wrapped up in a ROS Noetic node which has subscribers for end-effector target trajectory and robot state measurement. There is a publisher for the optimal control policy once computed. Additionally, there is a map sync module embedded with the MPC solver which allows a static world map to be uploaded to the solver. A Euclidean signed distance field [43] representation of the world will then be created within the solver as an additional constraint which allows obstacle avoidance based path planning.

The current state of the robot in Isaac-Sim is periodically published to the optimal control solver's MPC module. Note that the MPC module must have loaded the URDF of the robot it is to compute optimal trajectories for, in this case the Husky-KinovaGen3, beforehand. On receipt of the robot state, the solver computes an optimal policy and makes it available via a ROS topic which the robot is subscribed to. Isaac-Sim is set up as the model tracking node

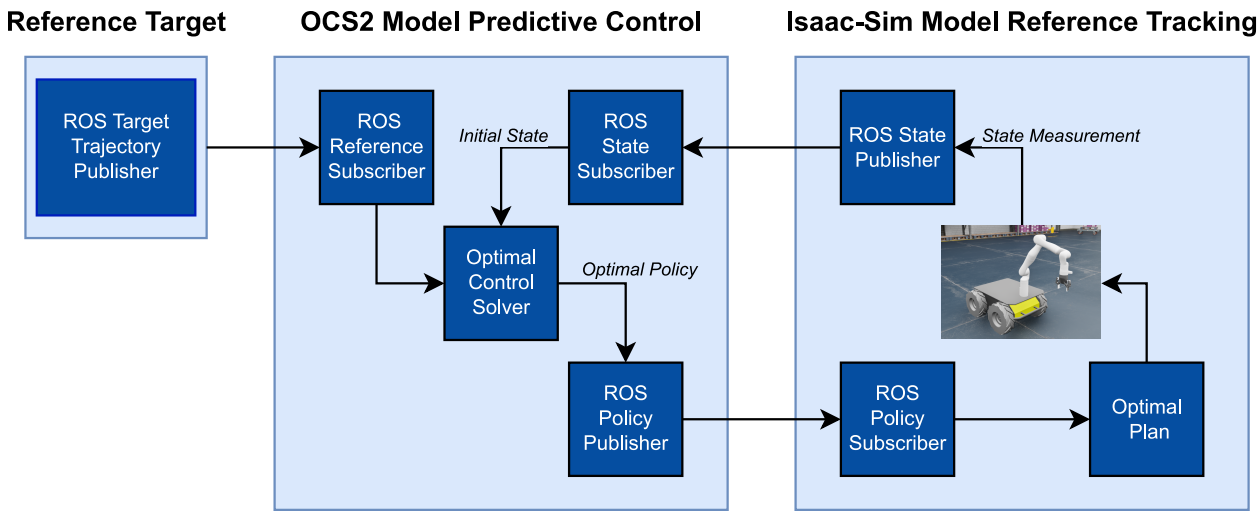


Figure 2.6: Diagram of how OCS2 was integrated with Isaac-Sim through ROS.

while OCS2 works on demand as the control node. This is online policy planning where the MPC sample time is determined by Isaac-Sim and the MPC computations are based solely on the robot's kinematics and most recent state measurement.

2.4 Vision

While OCS2 handles the gross movement of the Husky-KinovaGen3 in space, an additional piece is required for the accurate picking of any operator defined part in an industrial environment. Adding a vision node to the pick-and-place framework allows the operator to teach a prepick position above the part while capturing a reference image using the Kinova's built-in eye-in-hand camera. Once at the assigned prepick position, the hand camera is used to image the area and a vision application is leveraged for part localization, thus allowing accurate part picking.

It is important to note that research into the efficacy of different computer vision algorithms is beyond the scope of this work. All that is required for an effective pick-and-place framework is a vision system that is capable of localizing the part to be picked up by the manipulator. Since the proposed framework is vision system agnostic, others may use it to perform a larger study of vision methods.

2.4.1 Vision Implementation

The vision application developed for this work utilizes classic computer vision techniques for the identification and localization of the part. The eye-in-hand camera of the Kinova arm publishes both a point cloud image and an RGB image over designated ROS topics. This is done at a frame rate equal to that of the Isaac-Sim simulation. Additionally, the transform of the camera frame relative to the World frame is published over a ROS tf topic. By maintaining the relationship between coordinate frames, the ROS tf package keeps track of multiple frames over time, allowing simple transformation between points, vectors, etc., of any two frames at any point in time.

The vision node is implemented as a ROS service, which upon request will provide the precise location, in World coordinates, of the part to be picked up. The service is written with the C++ version of OpenCV[2], a fast and efficient computer vision library. The steps the service takes to calculate the part position are as follows:

1. Subscribe to the point cloud, RGB, and tf topics being published by the robot.

2. On receipt of an RGB image, first convert the image to grayscale, apply a Gaussian blur with a 3x3 kernel, then apply the Canny edge detection algorithm to extract the edges within the image. Sort all contours by area within the edge based image and find the centroid of the largest contour. This centroid gets stored as the center of mass (x,y position) of the part to be picked.

3. On receipt of a point cloud image, find the depth (the z coordinate) relative to the camera frame, of the stored x,y position of the part. This gives the x,y,z location of the part relative to the camera frame. Transform the x,y,z location from the camera frame to the World frame using the ROS tf library. Store the World relative x,y,z location of the part.

[2] opencv.org

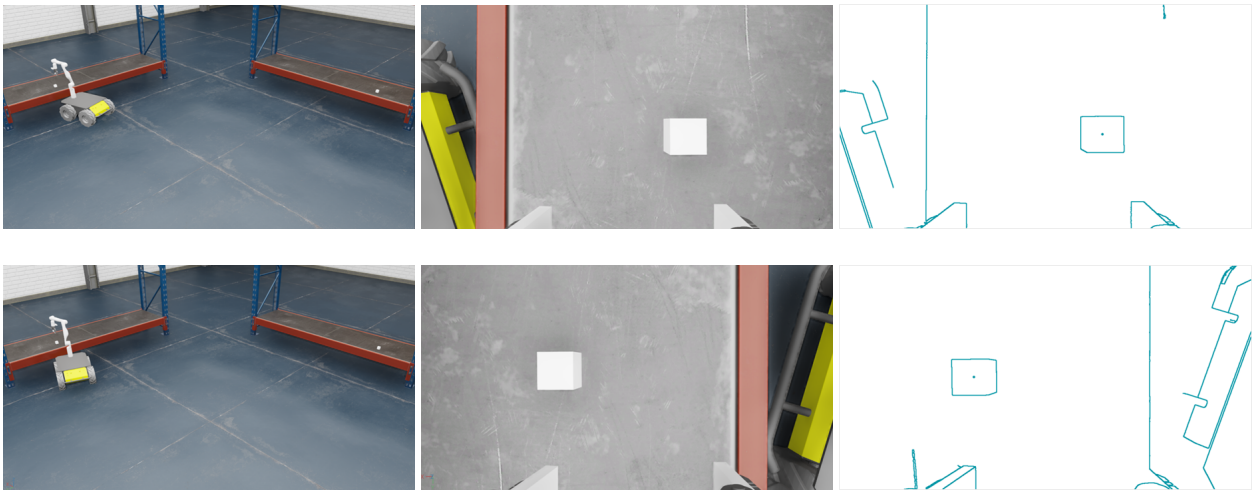


Figure 2.7: Example of the AIMM parking at and locating the part from two different configurations.

4. On a service request for part position, provide the latest x,y,z World relative location of the part.

As seen in Figure 2.2, the vision node is integrated into the rest of the pick-and-place framework via ROS. Figure 2.7 illustrates how the vision node allows an OCS2 controlled robot to pick a part regardless of base position or orientation. The top row left image shows the robot arriving at the part in one orientation. The RGB image is received by the vision service in the central picture. The picture on the right shows the Canny edge detected image with the part identified centroid. The row on the bottom illustrates the same workflow from a different base pose. As long as the end-effector has been driven to the prepick position and this position offers the eye-in-hand camera an image of the part, the vision provided service is called by the robot for the ascertainment of the precise x,y,z coordinates where the part is picked.

2.5 Communication and Containerization

As the proposed pick-and-place framework is comprised of three separate components, it is best practice that communication and application isolation frameworks be leveraged.

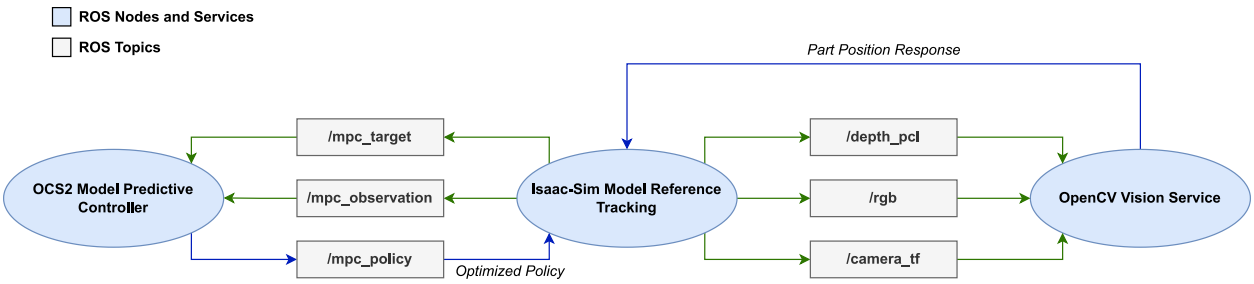


Figure 2.8: All ROS Noetic nodes and topics used in the framework.

As is seen in Figure 2.2, ROS Noetic[3] is utilized for communication among all three parts. ROS is a set of software libraries and tools such as drivers that help developers build robot applications. It is a publish/subscribe protocol which allows quick and efficient access to relevant robot data. Figure 2.8 shows all ROS topics utilized by each component in this work.

ROS provides default process isolation, and allows the creation of launch files for bringing up multiple services and nodes at once. However, as the pick-and-place framework is comprised of three separate pieces (vision, motion control, robot) that should be easy to swap for other similar services, an additional layer of isolation is used for this work. Docker "provides a set of service products that uses OS-level virtualization to deliver software in packages called containers"[4]. Once built, these container based applications are started, stopped and redeployed using Docker's built-in software tools. In this work, the ROS master, OCS2 MPC control node, and OpenCV vision node were each deployed as separate Docker containers which could be start/stopped together or individually, and which communicated with each other via ROS Noetic topics and services.

2.6 OCS2 Based Simulation Testbed

The Isaac-OCS2 testbed creates a safe environment within which the Husky-KinovaGen3's behavior under different cost and constraint functions is tested and evaluated. As the ef-

[3] noetic.org
[4] docker.com

ficacy of OCS2 for an Industry 5.0 based pick-and-place is to be tested, the simulation is based around a warehouse that can be customized, mapped, or quickly changed. The robot is loaded into the environment on start up of the application, and an environment file is provided with the testbed for setting parameters such as robot initial start postition/pose, friction parameters within the World, initial end-effector target, etc.

It is possible to start the simulation in either of two different modes. The first mode creates a target trajectory follow task. Moving the end-effector target goal within the UI sends a new reference target to OCS2, after which the MPC controller will begin driving the end-effector towards the new target. The second mode allows testing of the workflow in Figure 2.9. Assignment of a prepick position within Isaac-Sim represents the operator phase of the workflow. A manual trigger will then cause the robot to begin executing the navigation phase. The robot will drive to the prepick position, stop, image the part and request the 3D position from the vision service. Once the localized part position is received back from the service, a new end-effector target goal is sent to the MPC controller for moving to pick up the part. When the arm is in position, the gripper grabs the part, returns to the prepick position and proceeds to place the part at an operator predefined position. This placement phase is not explored in this work as it proceeds in the same manner as the pick phase (i.e., preplace $\rightarrow$ image $\rightarrow$ place $\rightarrow$ return home and wait for more parts).

2.7 Summary

Unlike traditional manufacturing production cells, the flexible manufacturing cell requires a mobile manipulator capable of performing a robust and flexible pick-and-place. Traditional docking based control techniques treat each robot as a separate entity and do not meet the challenge necessary for large-scale adoption of this technology in industry. This work proposes and implements a simulation testbed that uses a contemporary whole-body control, OCS2, to perform more flexible pick-and-place tasks. A mobile manipulator was created

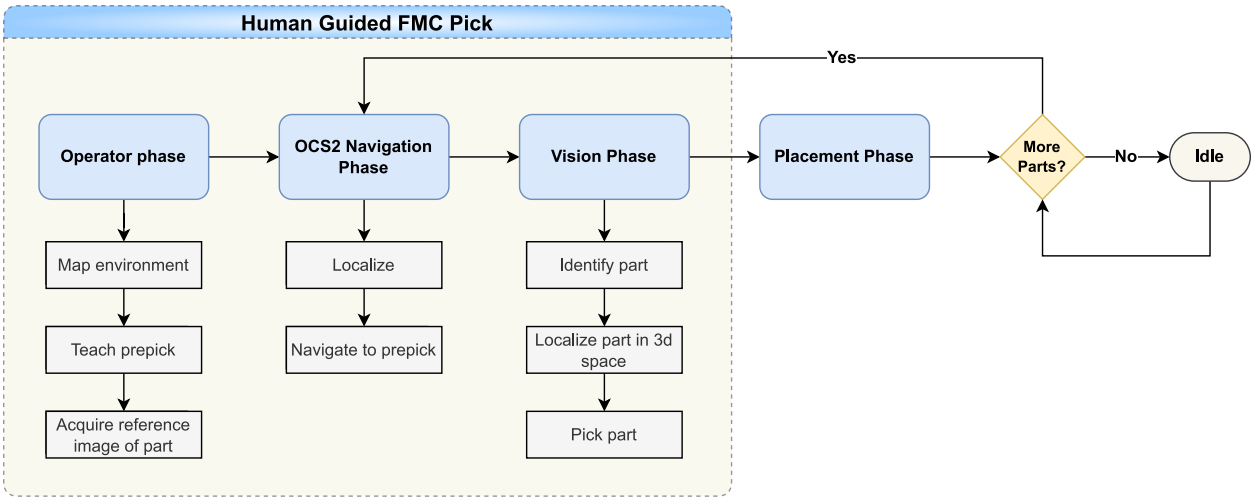


Figure 2.9: Diagram of the FMC pick-and-place workflow.

in the Isaac-Sim robotics simulator and OCS2 was used for command and control of the robot when driving to an operator assigned prepick position. A custom vision application is integrated for part locating and picking. ROS is used for communication between the disparate parts. This system supports the one-shot lead-through based assignment of a prepick position by an operator, thus enabling the cobot to drive to this position and successfully pick up the part agnostic of base orientation and/or position. The next chapter contains an evaluation of the developed framework, along with analysis of the robot's performance under a variety of cost and constraint values.

Chapter 3

System Evaluation

3.1 Overview

A major benefit of assessing robot planning frameworks in simulation is the ability to conveniently set up and/or change your robot's environment. The Industry 5.0 pick-and-place framework proposed here is intended to be utilized in an industrial environment/warehouse. It is assumed that this environment will have flat floors and be generally free of clutter. It is also assumed that the task to be performed by the AIMM involves repetitive picking of single or multiple objects above which the operator has previously taught the prepick positions. The goal of the system involves finding the optimal path from point A to point B, parking (agnostic of base pose) at the prepick point, locating and picking the part, placing it at another location, and then performing the task again. The goal of the testbed is the enablement of system performance evaluation within differing environments and under a variety of costs/constraints.

3.2 Testbed Use Case

Isaac-Sim provides several warehouse models that have various racks and shelves built into them. These models are useful because they are provided with collision meshes wrapped

around them. These collision meshes are how the simulator determines if objects are interacting in an appropriate (e.g., sitting on top of each other, being picked up) or inappropriate (e.g., colliding, crashing) manner. The imported robot also has this collision mesh wrapped around it. The built-in Isaac-Sim objects were utilized in the use case testbed as they satisfy the experimental needs while not overcomplicating the environment.

An image of the environment created for the use case is seen in Figure 3.1. This is a warehouse environment with a wide flat floor and two racks. There are parts to be picked on both the rack directly in front of the robot and the rack to the right of the robot. A prepick position has been assigned somewhere above each part from which the eye-in-hand camera can clearly image the part for picking. Note that these prepick positions are fluid, and are assigned in the testbed as desired. Similarly, the racks and objects to pick can also be moved around the floor. This is a basic supply chain management material handling task which Industry 5.0 cobots are ideally suited for. Routine and/or dangerous tasks such as packaging or heavy goods transportation are performed by the robot while the human is utilized in more complex jobs [7].

The goal of the use case is for the robot to drive to the prepick position of the part in front of it, image it, pick it, place it down again (as we are focused on the picking behavior), then proceed to the second part and do the same. It does this by minimizing the Euclidean distance and quaternion orientation between the end-effector and prepick positions. The test case is considered complete once the robot has successfully identified and picked both the first and second parts.

3.3 Robot Costs and Constraints

Evaluating the parking, picking and driving behavior of the robot at various input cost values offers insight into the efficiency and robustness of the system. Equation 2.1 defines the general form of the optimal control problem used in this work. The objective function

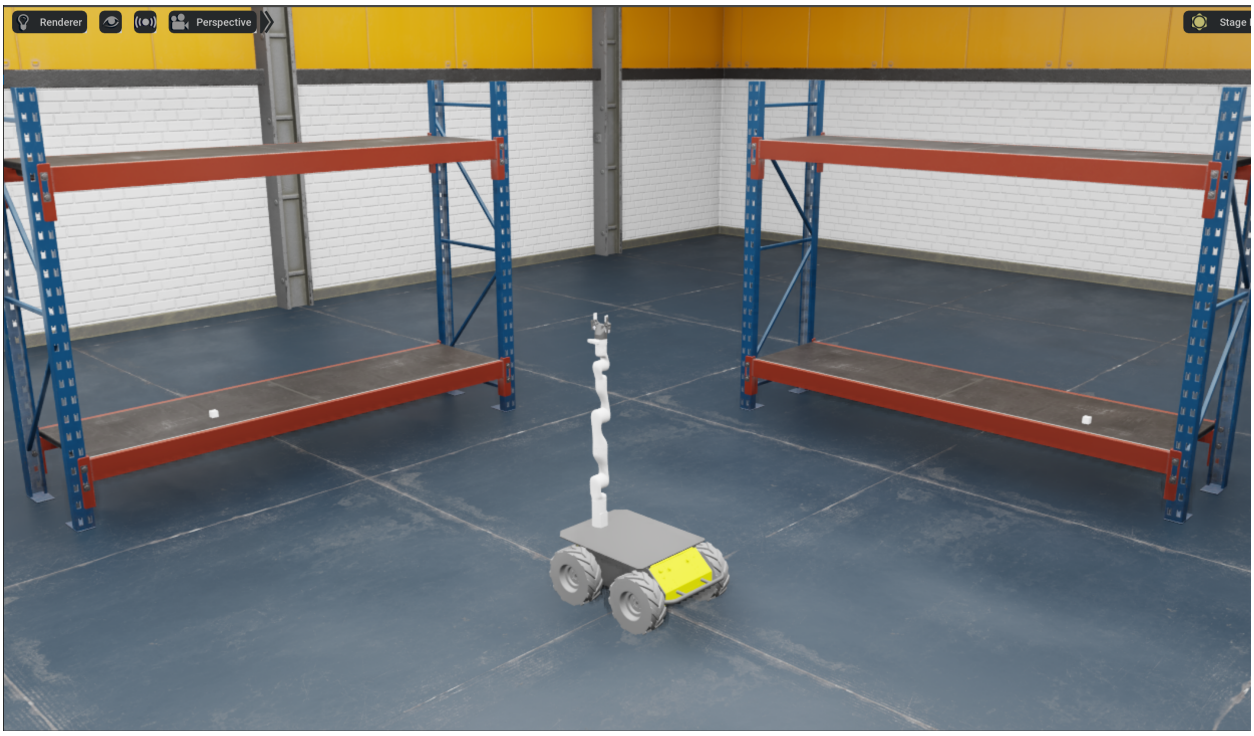


Figure 3.1: Isaac simulation testbed setup used for all plots.

$l(q(t), u(t), t)$ is a sum of five constraints/costs placed on the robot. These are:

1. A soft state-only equality constraint which penalizes the difference between desired and actual end-effector pose (position and orientation). This constraint contains additional penalty weights that the user can define. These factors allow increased weighting on deviation from the desired position/orientation.

2. A soft joint velocity limit constraint which seeks to keep the velocity of each DoF within a predefined upper and lower bound (e.g., $u_{min} \leq u \leq u_{max}$). All violations of this constraint are penalized at a value equal to the square of the violation.

3. A soft inequality self-collision constraint which seeks to ensure that the signed distance between the robot's rigid-body links stays positive. All violations of this constraint are equally penalized.

4. A soft state-only position constraint which seeks to keep the joints from exceeding user defined position limits (e.g., $q_{min} \leq q \leq q_{max}$). All violations of this constraint are

penalized at a value equal to the square of the violation.

5. A DoF input cost as defined in Equation 2.4. The cost matrix R defined in Equation 2.5 allows weighting of specific DoF inputs on the robot.

The joint position limits for each DoF were set as defined in the robot's associated URDF file. The joint velocity limits for each arm DoF were set as specified in the KinovaGen3 arm user manual (ranging from ± 1.0 rad/s). The forward velocity limit for the Husky base was set as ± 0.75 m/s. This is slightly less than the recommended velocity limit of ± 1.0 m/s, but experimentation in the simulator revealed that the robot maneuvered better at this slightly lower limit (perhaps due to the added weight of the Kinova arm). The angular velocity limit of the Husky base was set as the manual defined ± 2.0 rad/s.

The primary method of indirectly influencing the robot's movement is through tuning of the end-effector pose weights, and the robot input velocity cost weights. When setting the end-effector weights, it is important that the position weight be higher than the orientation weight. This is because it is desired that the robot drive as efficiently as possible to the target position. To do this, there should be a higher penalty on the end-effector position error than on the orientation error. This encourages the robot to prioritize the minimization of the Euclidean distance to the part, while only prioritizing the orientation error once the distance error has been appropriately minimized.

While it is possible to set the velocity input cost (i.e. the cost of movement) for each DoF individually, it is often more practical to differentiate between the cost of movement for the arm and the cost of movement for the base. Treating the robot in this way ensures that either the base or arm will lead the movement when seeking to reach the part. For a person, this is equivalent to the manner in which one would reach for a glass of water directly to their right. If there is a high cost of movement on the arm (e.g., one was holding packages in both arms), the person will turn their body before bending down to pick up the glass in one of their occupied arms. If there is a high cost of movement on the body (e.g., one was driving), the person will reach their arm to the right while barely turning in order to grab

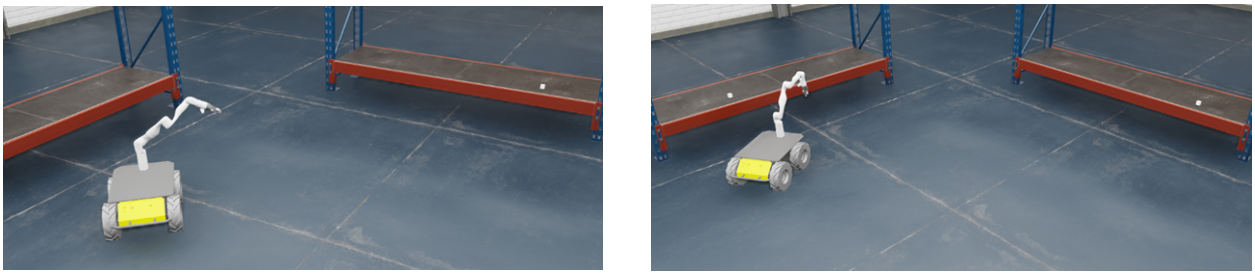


Figure 3.2: Turning comparison at 100x greater base cost (left image) vs 100x greater arm cost (right image).

the glass.

Figure 3.2 illustrates this principle as applied to the Kinova-HuskyGen3. In both images, the robot is in the midst of attempting to minimize its end-effector's distance to the part on the right. It must turn to the right in both situations, but due to differing movement costs placed on the robot, the manner in which it initiates the turn is different for each case. In the image on the left, the robot's cost of movement for the base is 100x the cost of movement for its arm. This encourages the solver to minimize the distance to the part by swinging the arm to right. In the image on the right, the exact opposite is true, as a 100x greater cost of movement for the arm encourages the solver to minimize the distance to the part by increasing the rate at which the robot base turns towards the part.

These differing movement costs also affect the manner in which the robot drives directly towards the part. Figure 3.3 presents the same comparison of 100x higher base cost versus 100x higher arm cost. In this scenario, the robot has finished turning and is driving on a direct line to the part. The leftmost image shows that at 100x higher base cost, the planner encourages the arm to stretch out as far as it can to reach the part. However, at 100x greater arm cost, the planner does not encourage this all out reach of the arm, but instead encourages the robot to drive just a bit faster.

Figures 3.4 through 3.8 provide a graphical view of the described behavior. These are the MPC computed base and arm velocities at differing costs as the robot executes the part picking use case previously described. From the graphs the following observations are made:

At a 10x greater arm than base cost, Figure 3.4, arm velocities do not exceed the

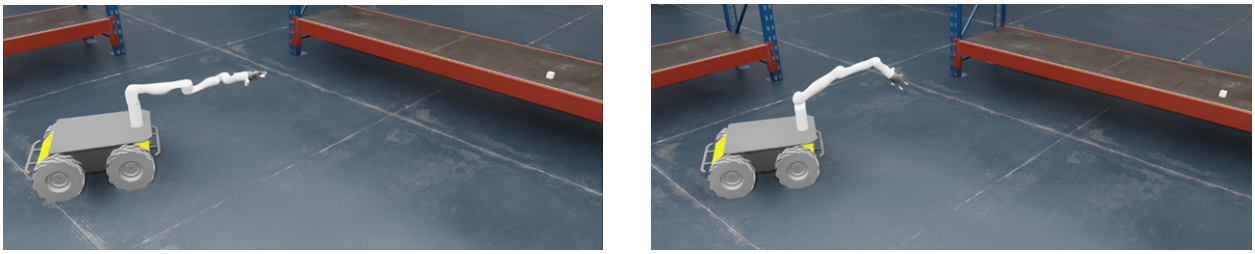


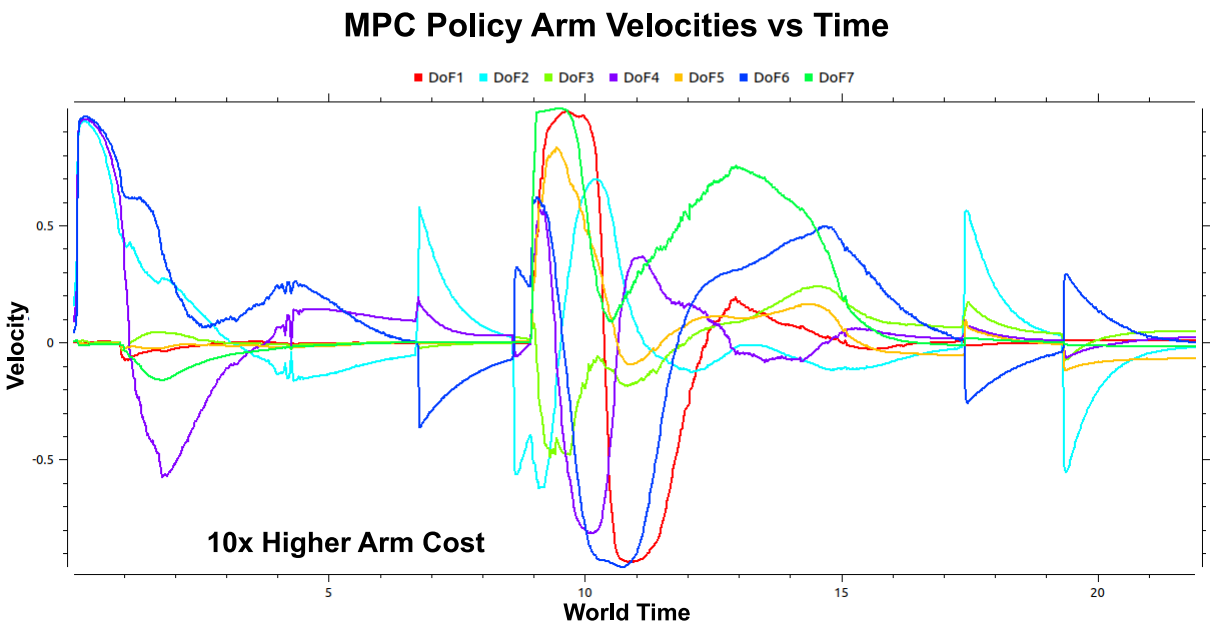
Figure 3.3: Driving comparison at 100x greater base cost (left image) vs 100x greater arm cost (right image).

±1.0 rad/s soft constraints. Linear base velocity temporarily exceeds the ±0.75 m/s soft constraint when turning to drive to the second part. Angular base velocity does not exceed the ±2.0 rad/s soft constraint. The linear velocity constraint violation is attributed to the lower base penalty for constraint violation balanced against the solver's need to minimize the position error to the part.

At 100x greater arm than base cost, Figure 3.5, arm velocities do not exceed the ±1.0 rad/s soft constraints. Linear base velocity temporarily exceeds the ±0.75 m/s soft constraint when approaching the second part. Angular base velocity temporarily exceeds the ±2.0 rad/s soft constraint when turning to drive to the second part. Note that of all the arm velocity graphs, this one shows the least amount of velocity "swing" for each DoF. Due to the much greater arm velocity cost, the solver violates both the angular and linear velocity constraints when trying to reach the part.

At 10x greater base than arm cost, Figure 3.6, arm velocities temporarily exceed the ±1.0 rad/s soft constraints when turning to go to the second part. Linear base velocity does not exceed the ±0.75 m/s soft constraint. Angular base velocity does not exceed the ±2.0 rad/s soft constraint. Note that the lower arm penalty encourages the solver towards solutions favoring greater arm movement.

At 100x greater base than arm cost, Figure 3.7, arm velocities do not exceed the ±1.0 rad/s soft constraints. Linear base velocity does not exceed the ±0.75 m/s soft constraint. Angular base velocity does not exceed the ±2.0 rad/s soft constraint. These cost settings appear to offer the greatest constraint adherence for this use case as the solver finds a nice



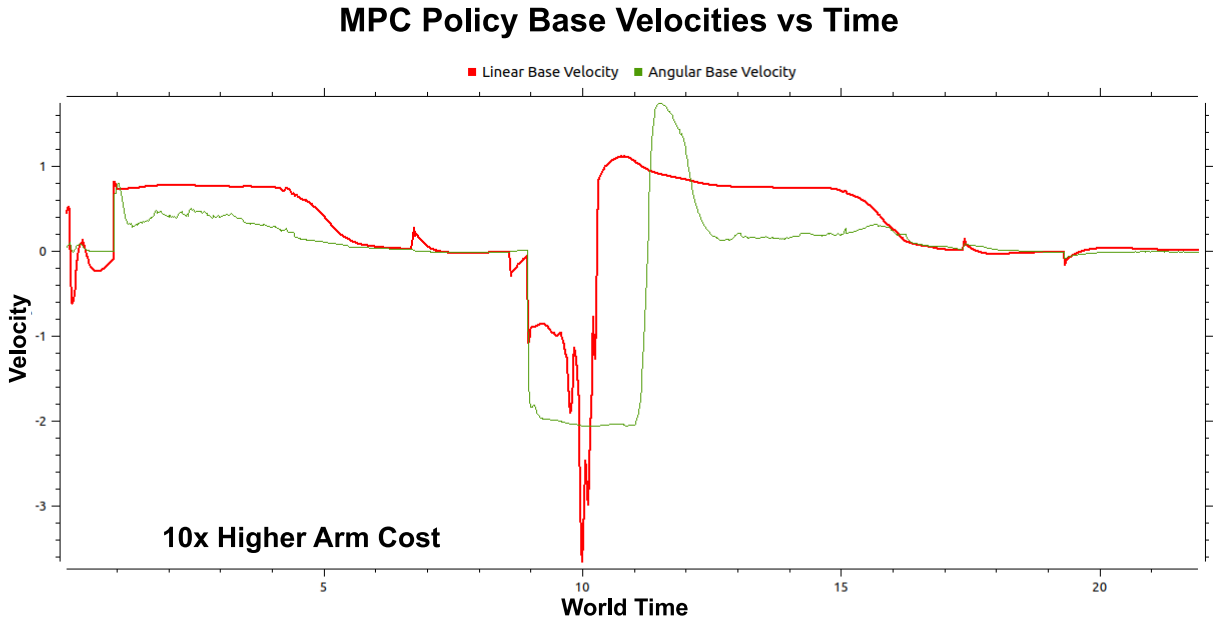


Figure 3.4: A comparison of MPC policy base and arm velocities at 10x arm cost.

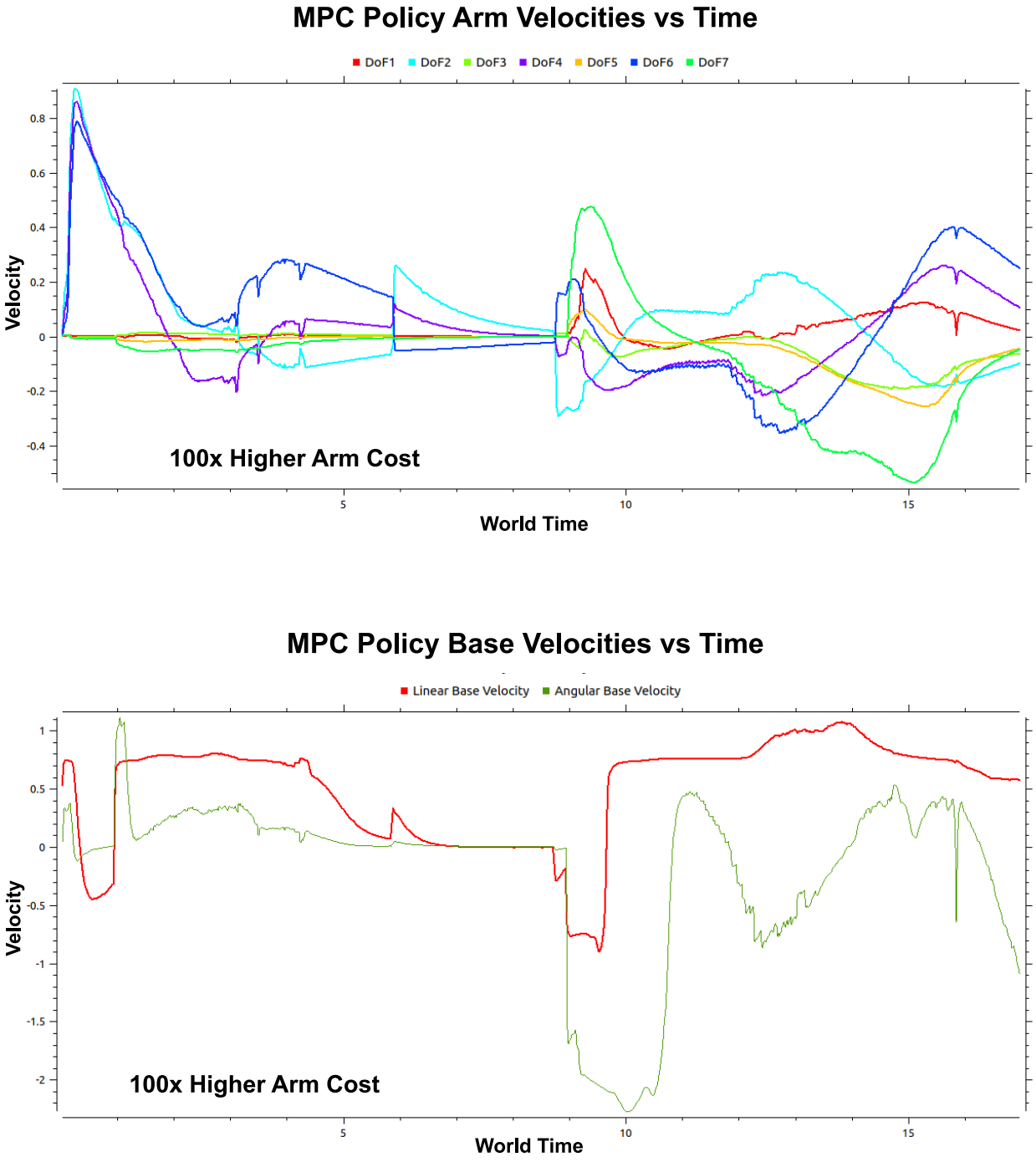


Figure 3.5: A comparison of MPC policy base and arm velocities at 100x arm cost.

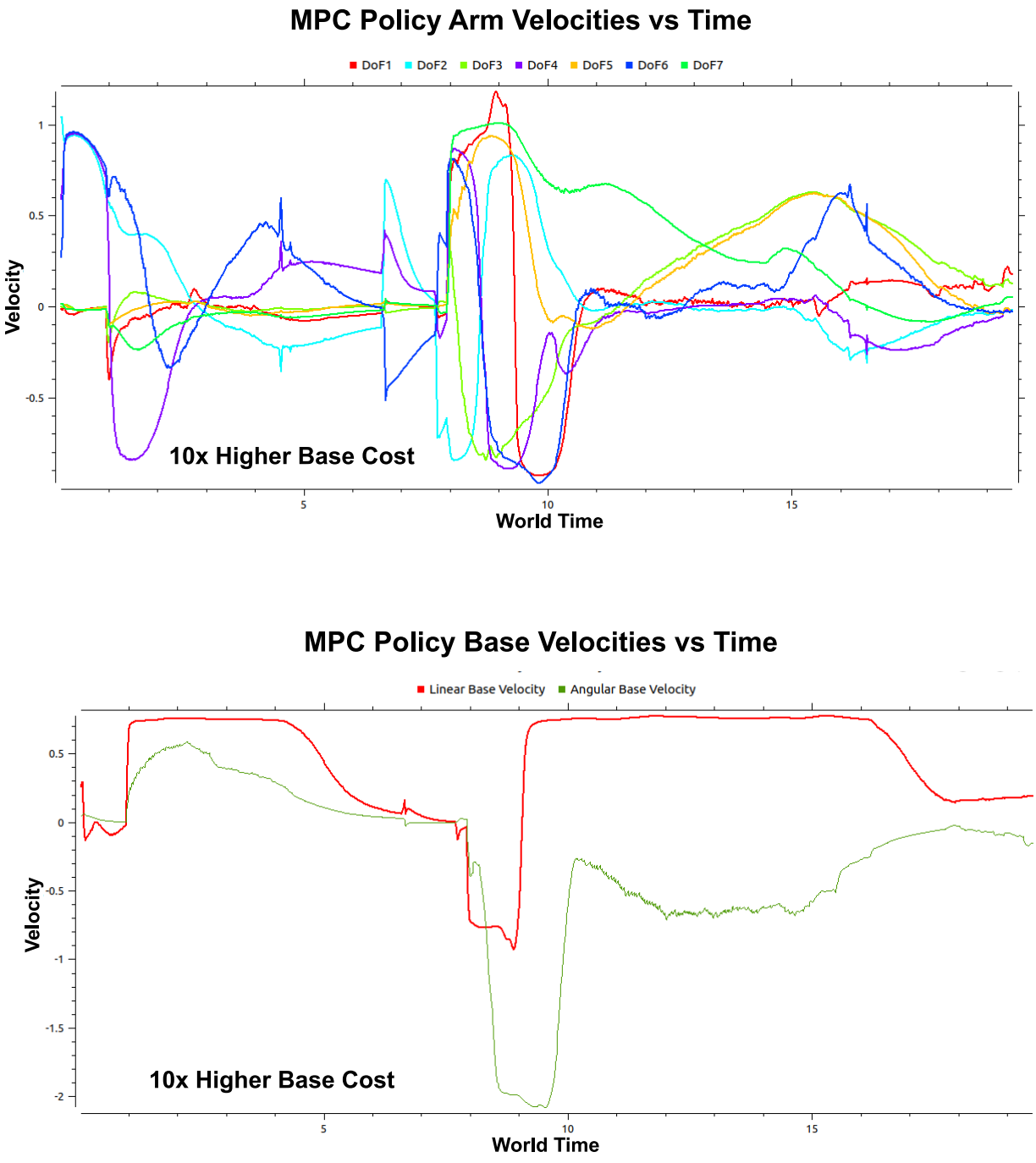


Figure 3.6: A comparison of MPC policy base and arm velocities at 10x base cost.

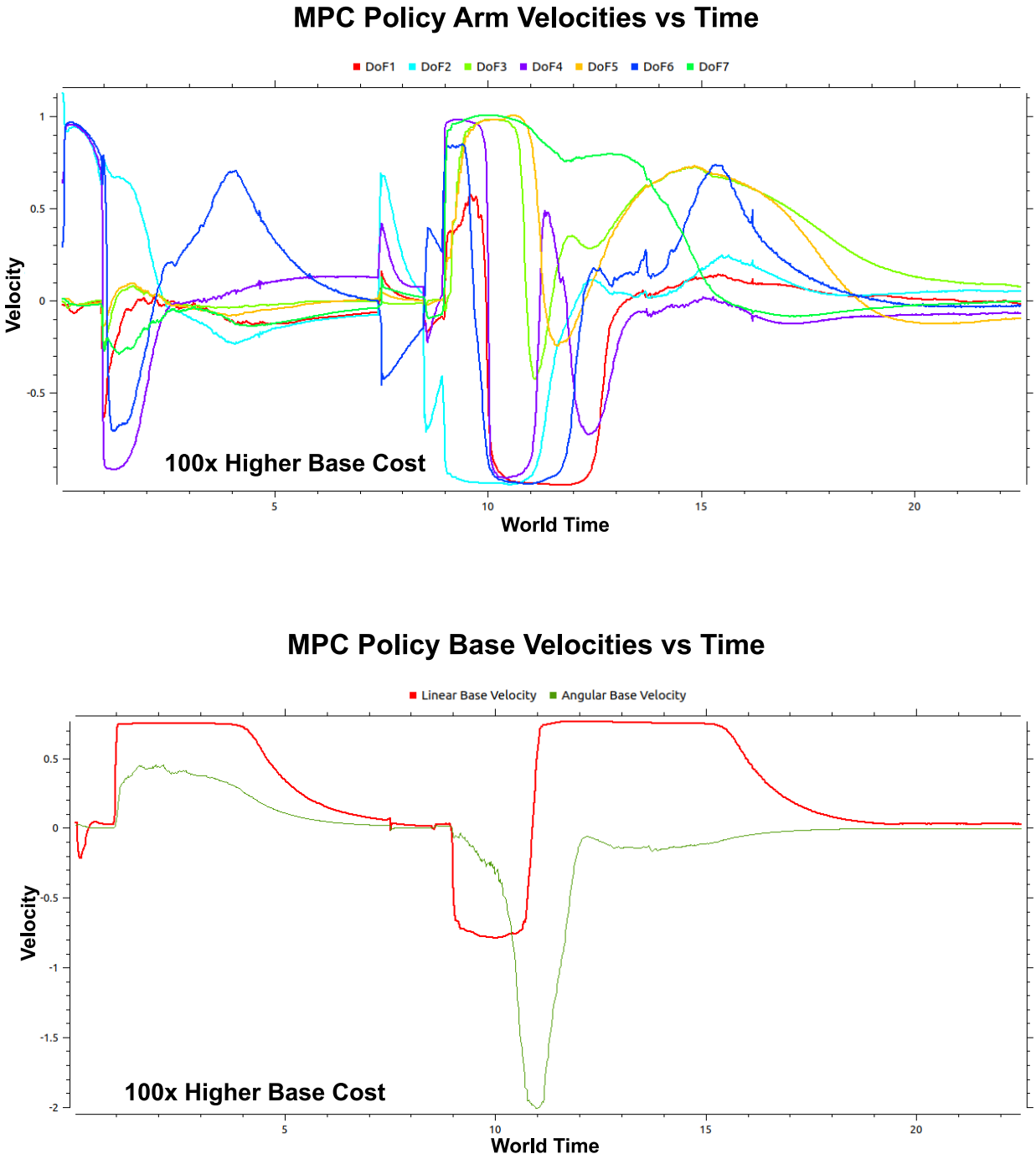


Figure 3.7: A comparison of MPC policy base and arm velocities at 100x base cost.

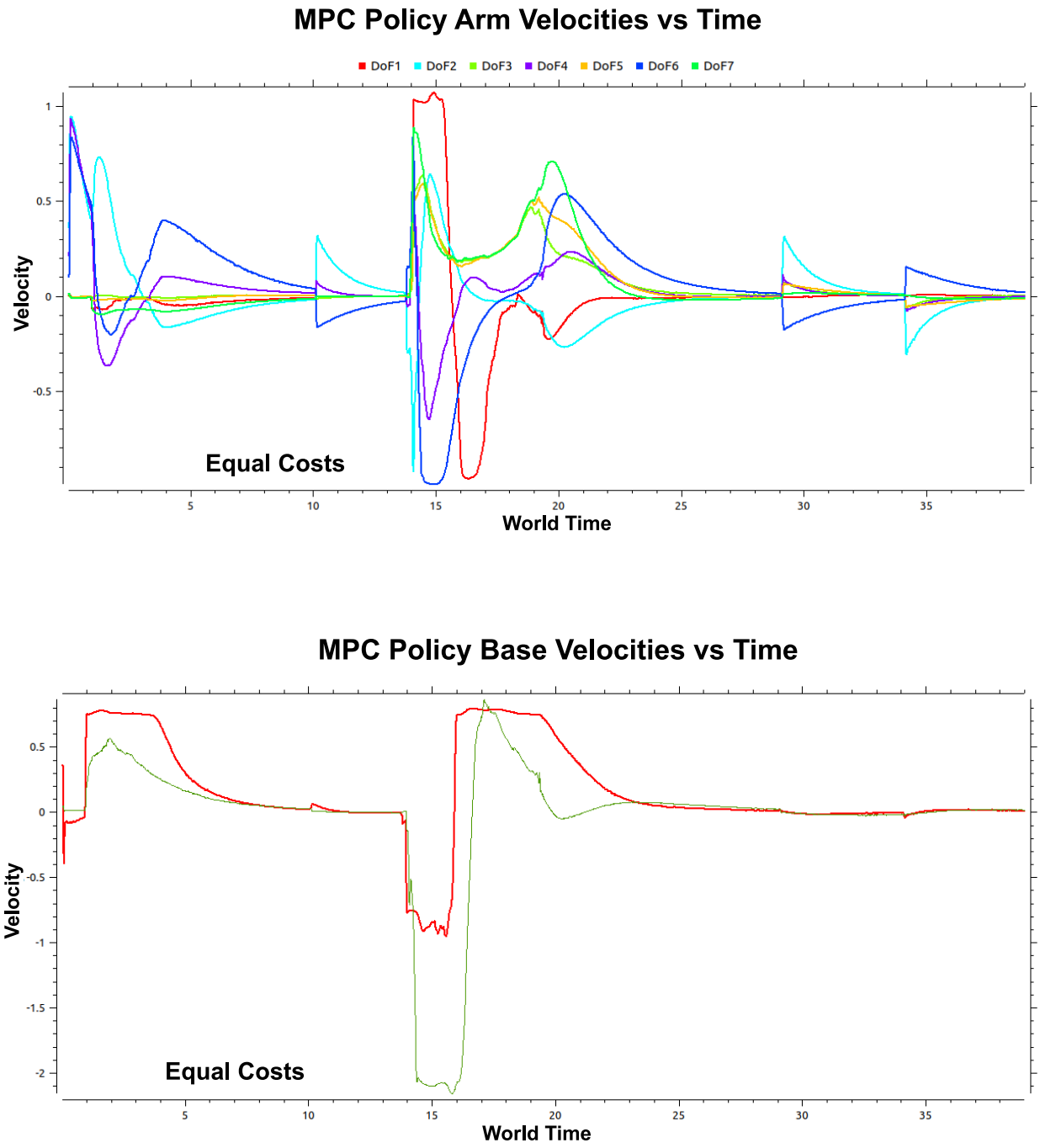


Figure 3.8: A comparison of MPC policy base and arm velocities at equal costs.

balance between position error minimization and velocity constraints.

At equal base and arm costs, Figure 3.8, arm velocities temporarily exceed the ±1.0 rad/s soft constraints when reaching for the second part. Linear base velocity does not exceed the ±0.75 m/s soft constraint. Angular base velocity temporarily exceeds the ±2.0 rad/s soft constraint when turning for the second part. Note that when neither the arm or base leads the movement, the manner in which the solver attempts minimization could involve violation of any and/or all constraints.

3.4 Robot Performance

For all tests in this work, the position error cost was set at a value of 2x the orientation error cost. This is the default value and seemed to work well as further increasing it had little perceptible effect on the robot's movement. However, as illustrated in Figures 3.4 through 3.8, adjusting the input-cost matrix R can have have a dramatic influence on the robot's optimized input trajectory.

When the base or arm costs are set such that one or the other will be the "leader", the robot is able to repeatedly and successfully execute the use case in an efficient manner. Figure 3.9 illustrates the robot's base and end-effector paths in space while executing the use case. It is seen that the robot efficiently drives to and picks the first part. It then executes an efficient reverse while turning maneuver before proceeding along a linear path to the second part.

The robot does not always perform as well when the arm and base costs are set equal to each other. While often able to perform the whole use case, there were occasions where it appeared that the MPC solver computed solution collided with the Isaac-Sim PhysX engine in such a way that the robot began spinning uncontrollably in circles. The reason for this behavior is not entirely known, as it is not dynamically feasible in the real world. It is speculated that the unpredictability of the solver's solution when both the base and arm

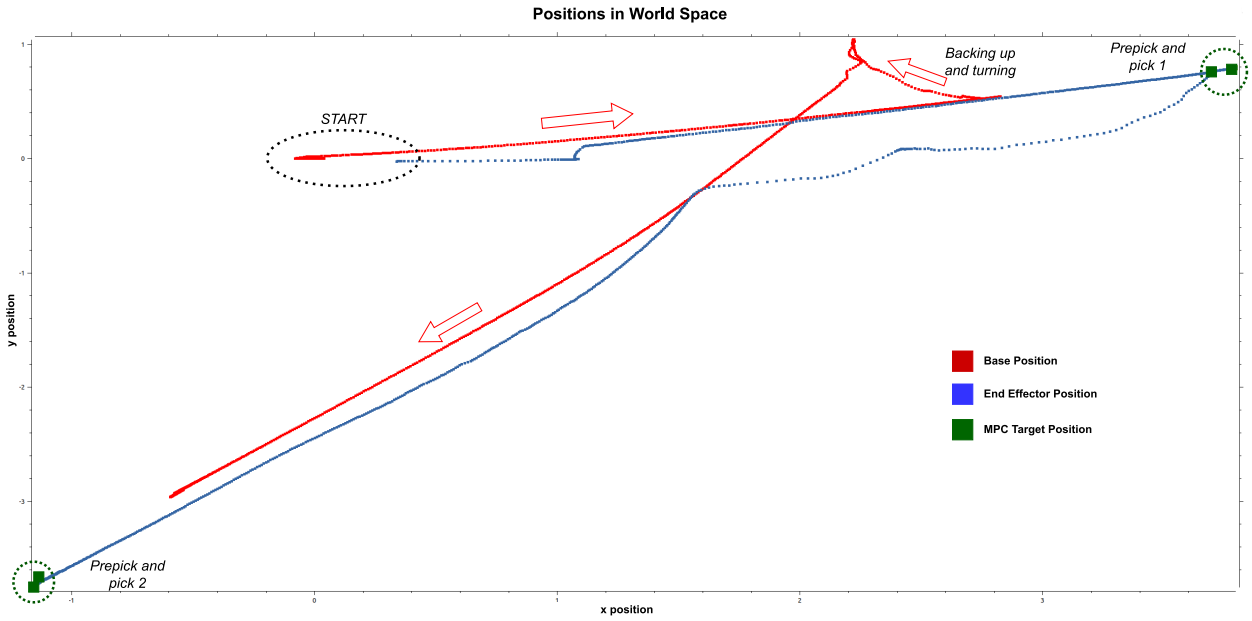


Figure 3.9: Base and end-effector positions in x,y world space during picking task. This run was performed with a 10x higher arm than base cost.

constraints are violated sometimes leads to a collision with PhysX when a proposed solution is not "dynamically" possible within simulation.

Repeated simulation runs demonstrate the velocity controlled robot smoothly following the MPC solver computed input trajectories. There is no obvious jittering, overshoot, etc. When the end-effector reaches the target position, it is capable of maintaining that position via velocity inputs from the solver. A closer look at the robot's actual velocity vs the MPC proposed velocity is shown in Figure 3.10.[1] This reveals that the robot arm and base velocities generally track with those computed by the solver. However, there is much more noise in the robot's actual velocities, especially when compared against the smoothness of the solver computed velocities. This could potentially be due to the friction of the robot DoFs needing to be increased in simulation so that a sudden velocity increase of one joint does not directly affect the others.

[1]See Appendix A for additional plots of policy vs observation at different cost values.

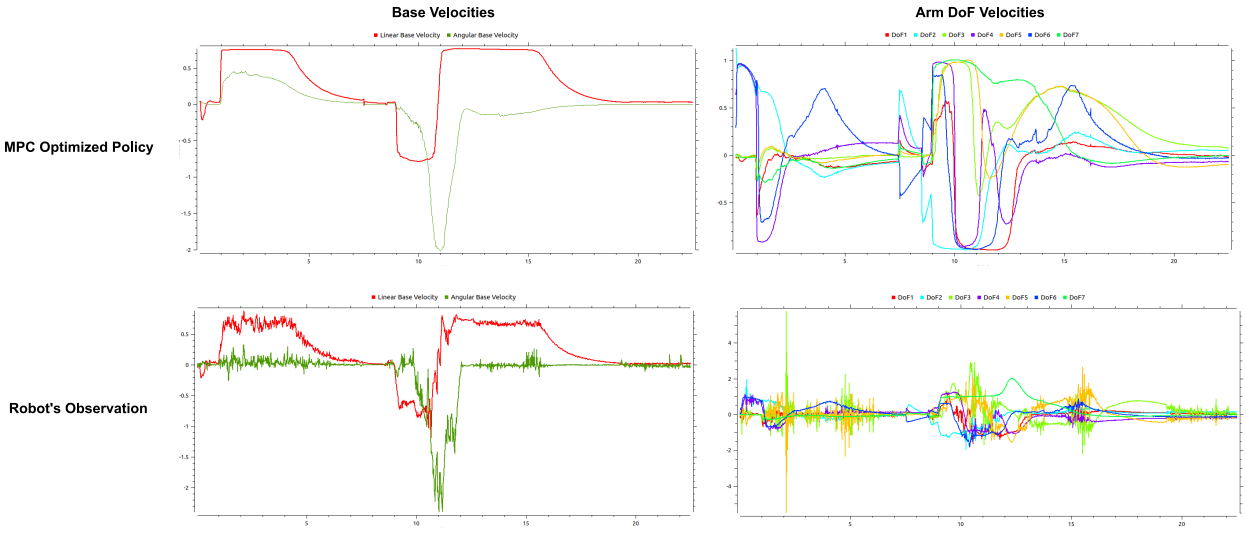


Figure 3.10: A comparison of the optimized MPC policy vs the robot's observation of that applied policy.

3.5 Summary

The robot's trajectory planning is heavily influenced by the input associated cost values. Experimentation, tuning, and use case assessment are all vital before placing any robot onto a production floor. Understanding how the robot will respond to various control variables is a requirement for a safe deployment. If it is desired that the robot be capable of quickly reaching for parts within close proximity of each other then the arm cost terms should be set well below those of the base. However, if many people are around, it may be desirable that the robot focus on moving the base as much as possible when transitioning between stations. The developed testbed enables the creation of custom environments within which a modern, Industry 5.0 supported, whole-body controlled pick-and-place is tested and evaluated.

Compared to docking based methods, the whole-body controlled pick-and-place framework provides better flexibility to an Industry 5.0 cell. In many scenarios, it is not always feasible to mount a docking station at every location you wish the robot to be able to park at and pick objects from. Reconfiguration of the cell would require unmounting the docking stations and then remounting them at the new desired pick points. By allowing the operator to assign prepick positions which the robot can drive to and park at without regard to

base pose, the overall robustness of the 5.0 operation is improved. Additional benefits of the developed system include the adaptability to various vision systems and the potential for different cobots to be modeled and tested before they are purchased and deployed to the factory floor. This cobot flexibility could lead to cost savings if it is discovered that a particular AIMM configuration does not perform well under certain environment conditions and/or constraints.

Chapter 4

Conclusion

4.1 Conclusion

This work proposes an Industry 5.0 compliant pick-and-place framework for improving the robustness and flexibility of Autonomous Industrial Mobile Manipulators on the production floor.

The central questions for this research were as follows:

1. Could this system support the one-shot lead-through based assignment of a prepick position by an operator, thus enabling the mobile manipulator to drive to this position and successfully pick up the part agnostic of base orientation and/or position?
2. Does this system perform the task faster (defined as amount of time taken to undock $\rightarrow$ drive $\rightarrow$ dock $\rightarrow$ pick) than docking based methods?

Rather than the currently popular docking based methods, the framework leverages a modern, whole-body, model predictive control method (OCS2) for the robot's gross motion planning. A computer vision based object identification algorithm for efficient locating and picking of a desired part was also created. The vision component is plugable and easily modified for the support of various part locating algorithms. The vision and planner components are networked through ROS and integrated with an Isaac-Sim based mobile

manipulator. The vision, planner, and simulator comprise a testbed within which the pick-and-place framework's utility at various cost and constraint optimization values can be safely explored and assessed.

Testing and evaluation in the testbed demonstrated that, agnostic of base pose and/or orientation, the pick-and-place framework supports the assignment, identification and picking of an operator defined part. The removal of the docking constraint for the base when navigating to a goal position enables the robot to approach and park at the operator defined prepick position in a number of different ways. Unlike docking based methods which require considerable time to locate and dock, the whole-body control utilized in this work enables a much more efficient parking procedure. In conjunction with the plugable vision system, the pick-and-place framework keeps the human in the loop of the robot's control, while allowing just enough autonomy that the robot is capable of intelligently performing the same repetitive task in a safe, robust, flexible, and efficient manner.

4.2 Future Work

A primary focus of future research should include deployment of the developed system to a live Husky-KinovaGen3 mobile robot. Utilizing the work done here, the costs and constraints can be tuned in simulation before deployed in reality. Such a deployment would enable testing and evaluation of the simulated system's performance as compared to the real one. Additionally, different vision algorithms could be tuned and deployed to the real-world system. These vision programs could leverage techniques for more advanced reference imaging and part identification in object rich environments, or even develop and train a vision based machine learning model for part picking.

Additional research could include the development of a front-end based API interface to the system for switching the robot into different modes of operation. Possibilities include: operator tuning of cost/constraint weights, switching the robot between end-effector and

base tracking tasks, updating environmental mappings, reassigning prepick positions, etc.

Appendix A

Additional Figures

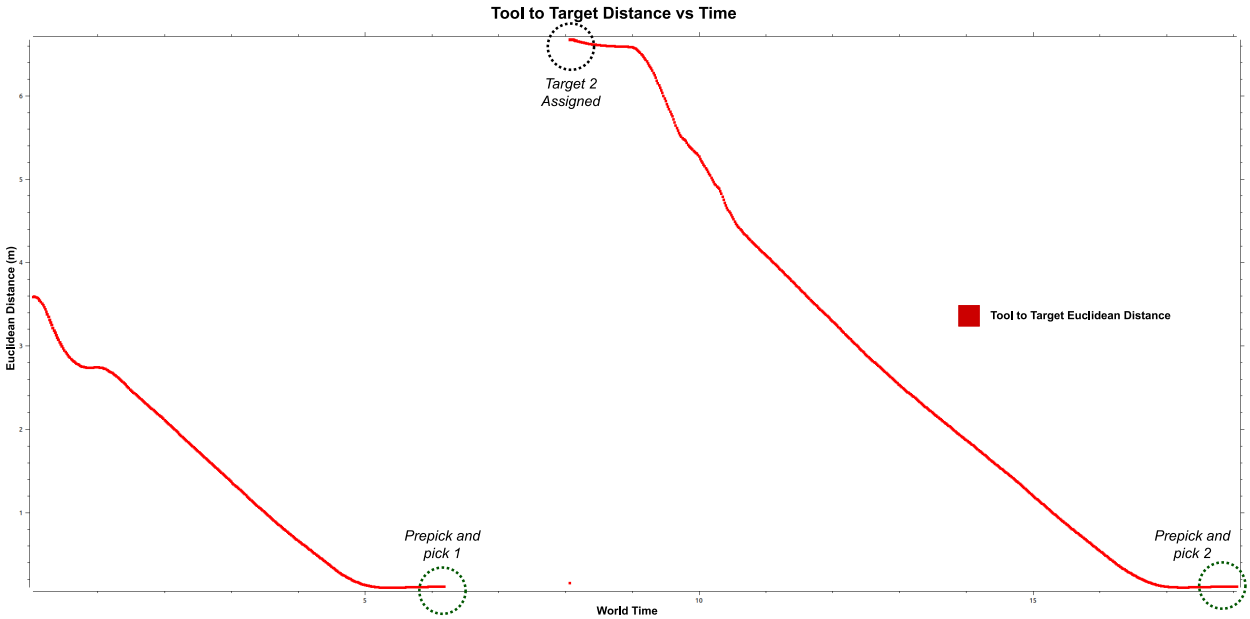


Figure A.1: Example of OCS2 minimizing the end-effector distance to prepick and pick 1 followed by prepick and pick 2.

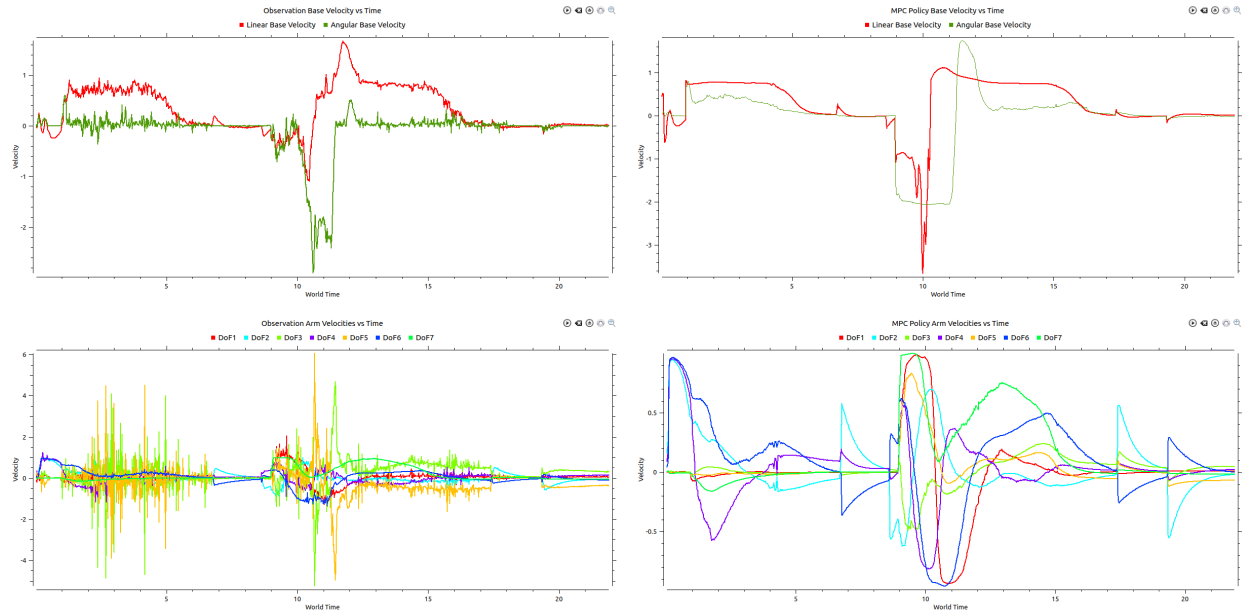


Figure A.2: 10x higher arm than base cost encourages base movement when trying to reach the part. Graphs on left are the robot's observation velocities. Graphs on right are the first optimized input as computed by the MPC solver.

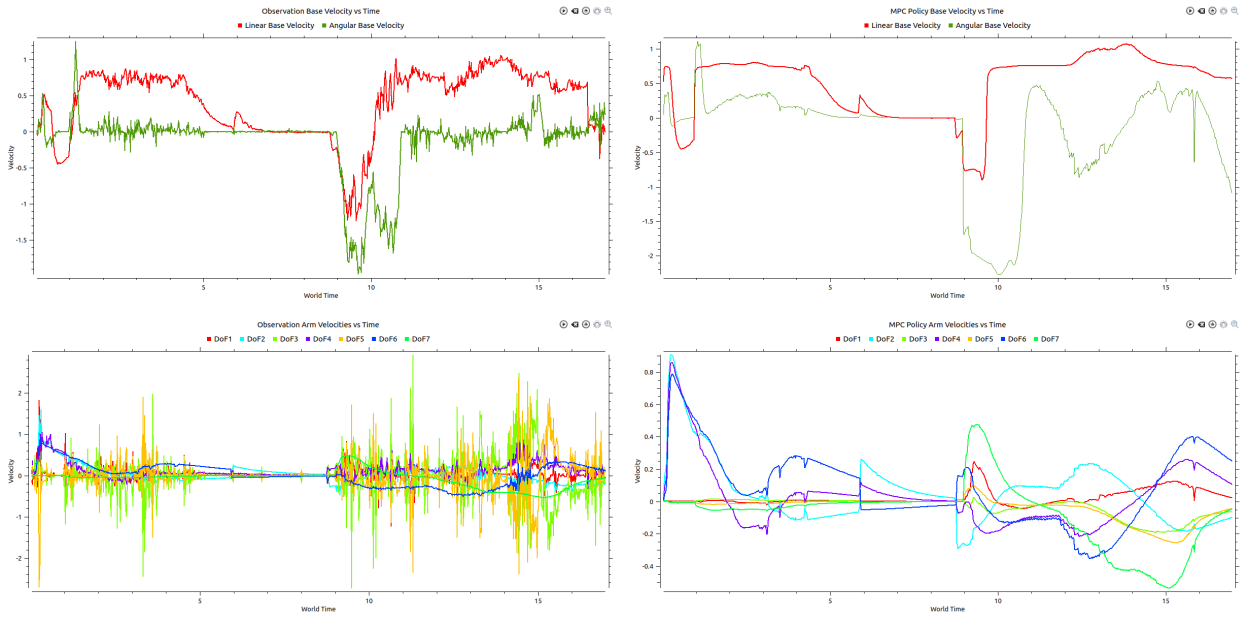


Figure A.3: 100x higher arm than base cost further encourages base movement when trying to reach the part. Graphs on left are the robot's observation velocities. Graphs on right are the first optimized input as computed by the MPC solver.

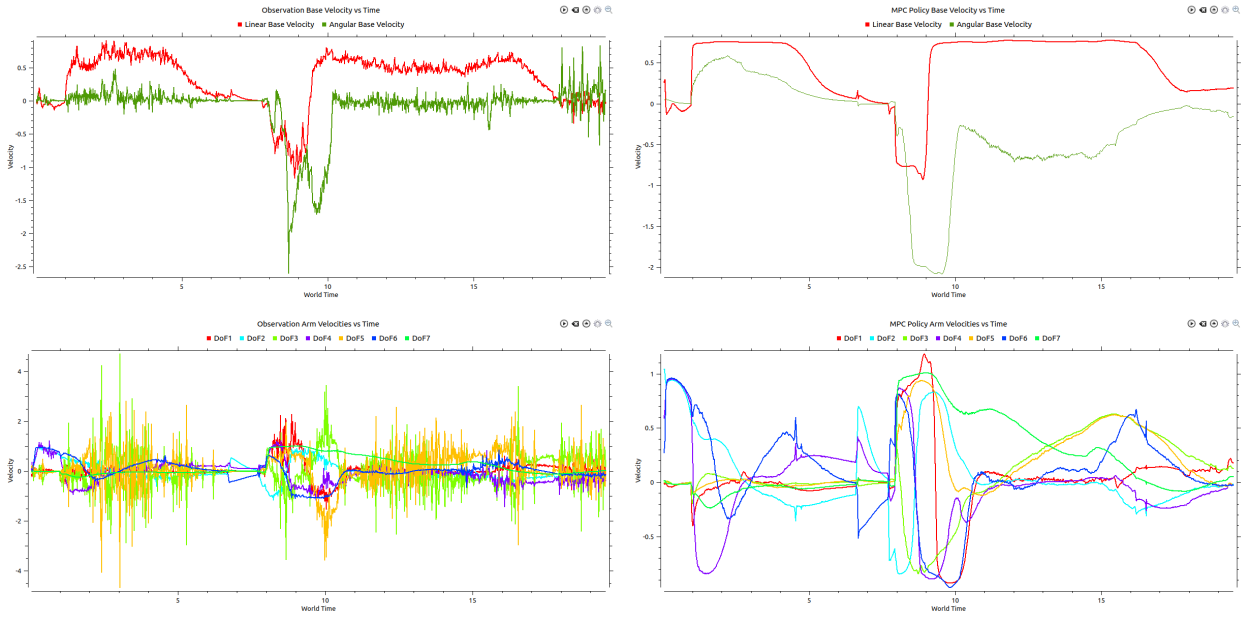


Figure A.4: 10x higher base than arm cost encourages arm movement when attempting to minimize the distance to the part. Graphs on left are the robot's observation velocities. Graphs on right are the first optimized input as computed by the MPC solver.

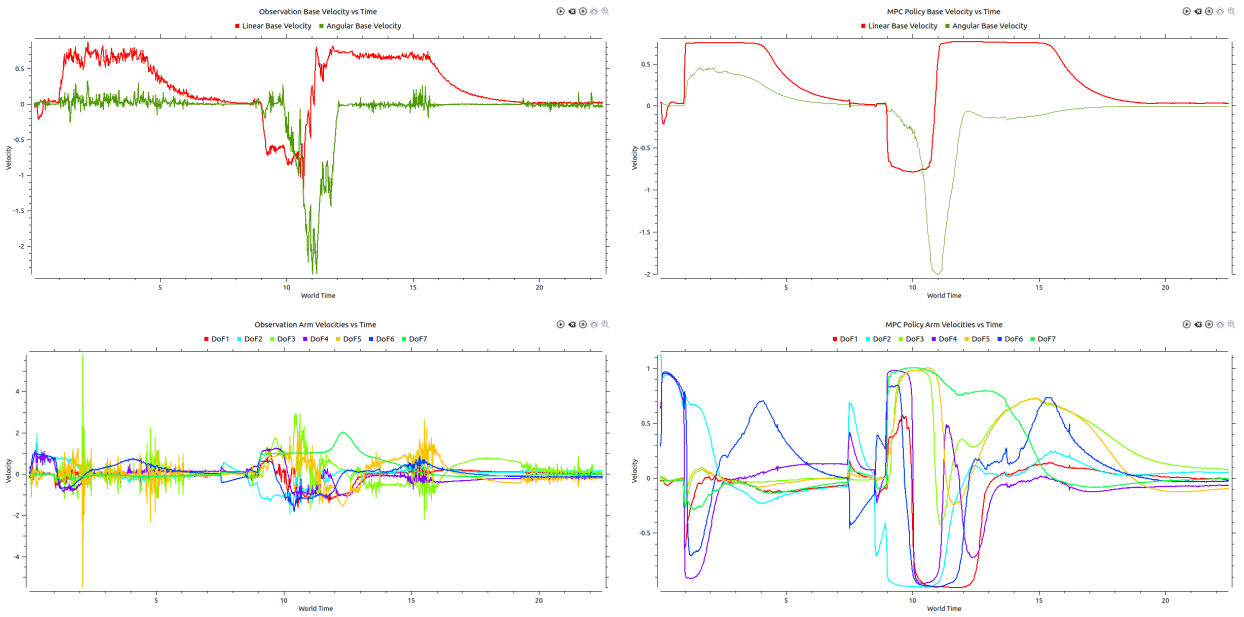


Figure A.5: 100x higher base than arm cost further encourages arm movement when attempting to minimize the distance to the part. Graphs on left are the robot's observation velocities. Graphs on right are the first optimized input as computed by the MPC solver.

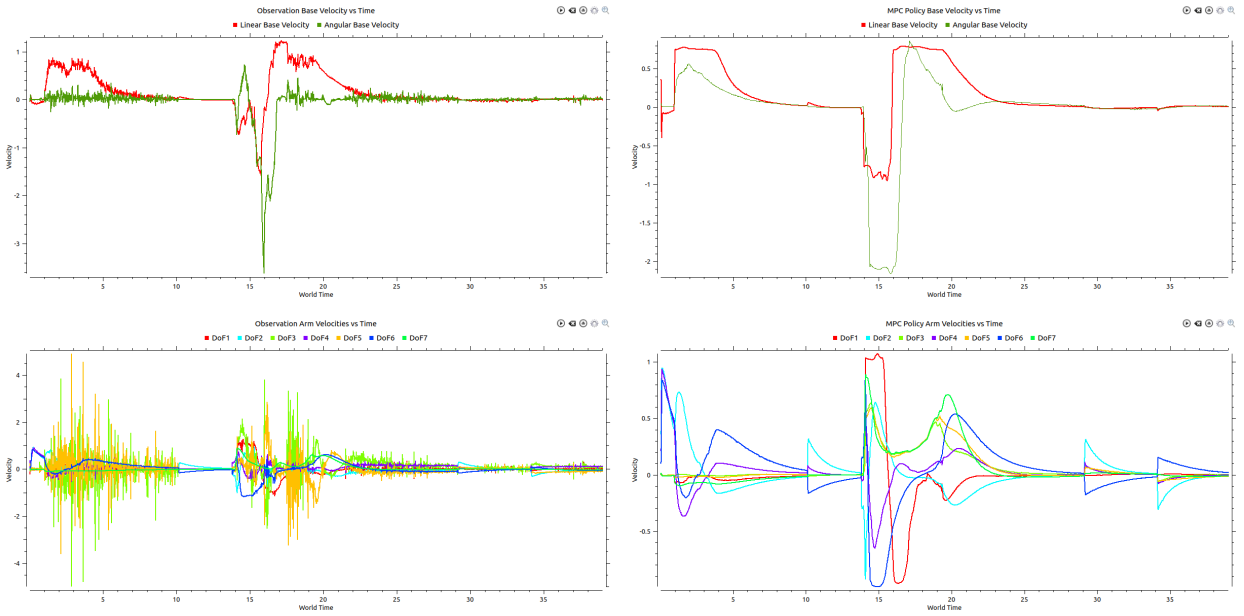


Figure A.6: Velocities when the arm and base costs are equal to each other. Graphs on left are the robot's observation velocities. Graphs on right are the first optimized input as computed by the MPC solver.

Vita

With an initial degree in Mechatronics Technology from Tidewater Community College, Ethan has worked many years in the industrial manufacturing world. Working first as a Machine Operator and then a Mechatronics Technician, he was exposed to a variety of Industry 4.0 technologies and methods. He worked full time for six years while taking two classes per semester to earn his undergraduate degree in Computer Engineering from Virginia Commonwealth University. He has worked the past three years at the Commonwealth Center for Advanced Manufacturing (CCAM) as first a Computer Engineering intern and then, while on a VCU-CCAM fellowship, a Software Development Graduate Research Assistant. His interests lie in system integration, software development, edge/embedded systems, auto repair, and sports.

References

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[2] A. G. Frank, L. S. Dalenogare, and N. F. Ayala, "Industry 4.0 technologies: Implementation patterns in manufacturing companies," *International Journal of Production Economics*, vol. 210, pp. 15–26, 2019.

[3] L. S. Dalenogare, G. B. Benitez, N. F. Ayala, and A. G. Frank, "The expected contribution of industry 4.0 technologies for industrial performance," *International Journal of production economics*, vol. 204, pp. 383–394, 2018.

[4] T. Margaria and A. Schieweck, "The digital thread in industry 4.0," in *International Conference on Integrated Formal Methods*. Springer, 2019, pp. 3–24.

[5] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, "Industry 4.0 and industry 5.0—inception, conception and perception," *Journal of Manufacturing Systems*, vol. 61, pp. 530–535, 2021.

[6] L. Alexa, M. Pîslaru, and S. Avasilcăi, "From industry 4.0 to industry 5.0—an overview of european union enterprises," *Sustainability and Innovation in Manufacturing Enterprises*, pp. 221–231, 2022.

[7] P. K. R. Maddikunta, Q.-V. Pham, B. Prabadevi, N. Deepa, K. Dev, T. R. Gadekallu, R. Ruby, and M. Liyanage, "Industry 5.0: A survey on enabling technologies and po-

tential applications," *Journal of Industrial Information Integration*, vol. 26, p. 100257, 2022.

[8] S. Zhang, S. Li, H. Wang, and X. Li, "An intelligent manufacturing cell based on human–robot collaboration of frequent task learning for flexible manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 120, no. 9, pp. 5725–5740, 2022.

[9] A. Adel, "Future of industry 5.0 in society: human-centric solutions, challenges and prospective research areas," *Journal of Cloud Computing*, vol. 11, no. 1, pp. 1–15, 2022.

[10] H. Arnarson and B. Solvang, "Reconfigurable autonomous industrial mobile manipulator system," in *2022 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2022, pp. 772–777.

[11] S. Bøgh, M. Hvilshøj, M. Kristiansen, and O. Madsen, "Identifying and evaluating suitable tasks for autonomous industrial mobile manipulators (aimm)," *The International Journal of Advanced Manufacturing Technology*, vol. 61, no. 5, pp. 713–726, 2012.

[12] M. Hvilshøj, S. Bøgh, O. S. Nielsen, and O. Madsen, "Autonomous industrial mobile manipulation (aimm): past, present and future," *Industrial Robot: An International Journal*, 2012.

[13] S. KuCuk and Z. Bingul, "The inverse kinematics solutions of industrial robot manipulators," in *Proceedings of the IEEE International Conference on Mechatronics, 2004. ICM'04.* IEEE, 2004, pp. 274–279.

[14] S. Choi, W. Eakins, G. Rossano, and T. Fuhlbrigge, "Lead-through robot teaching," in *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, 2013, pp. 1–4.

[15] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, 2021.

[16] A. C. Acosta Calderon, B. S. Ng, E. R. Mohan, and H. K. Ng, "Docking system and power management for autonomous mobile robots," in *Applied Mechanics and Materials*, vol. 590. Trans Tech Publ, 2014, pp. 407–412.

[17] M. M. Costa and M. F. Silva, "A survey on path planning algorithms for mobile robots," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2019, pp. 1–7.

[18] Y. Dai, C. Xiang, Y. Zhang, Y. Jiang, W. Qu, and Q. Zhang, "A review of spatial robotic arm trajectory planning," *Aerospace*, vol. 9, no. 7, p. 361, 2022.

[19] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter, "Whole-body mpc for a dynamically stable mobile manipulator," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3687–3694, 2019.

[20] M. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg, "Articulated object interaction in unknown scenes with whole-body mobile manipulation," *arXiv preprint arXiv:2103.10534*, 2021.

[21] A. AlAttar, D. Chappell, and P. Kormushev, "Kinematic-model-free predictive control for robotic manipulator target reaching with obstacle avoidance," *Frontiers in Robotics and AI*, vol. 9, 2022.

[22] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[23] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[25] P. Wu, A. Escontrela, D. Hafner, K. Goldberg, and P. Abbeel, "Daydreamer: World models for physical robot learning," *arXiv preprint arXiv:2206.14176*, 2022.

[26] M. A. Lee, C. Florensa, J. Tremblay, N. Ratliff, A. Garg, F. Ramos, and D. Fox, "Guided uncertainty-aware policy optimization: Combining learning and model-based strategies for sample-efficient policy learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7505–7512.

[27] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6023–6029.

[28] P. Weinzaepfel, R. Brégier, H. Combaluzier, V. Leroy, and G. Rogez, "Dope: Distillation of part experts for whole-body 3d pose estimation in the wild," in *European Conference on Computer Vision*. Springer, 2020, pp. 380–397.

[29] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[30] Y. Shi, Z. Chen, Y. Wu, D. Henkel, S. Riedel, H. Liu, Q. Feng, and J. Zhang, "Combining learning from demonstration with learning by exploration to facilitate contact-rich tasks," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1062–1069.

[31] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[32] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A survey on learning-based robotic grasping," *Current Robotics Reports*, vol. 1, no. 4, pp. 239–249, 2020.

[33] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[34] X. Sun, X. Zhu, P. Wang, and H. Chen, "A review of robot control with visual servoing," in *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 2018, pp. 116–121.

[35] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[36] F. Farshidian *et al.*, "OCS2: An open source library for optimal control of switched systems," [Online]. Available: https://github.com/leggedrobotics/ocs2.

[37] S. Bøgh, M. Hvilshøj, M. Kristiansen, and O. Madsen, "Autonomous industrial mobile manipulation (aimm): from research to industry," in *Proceedings of the 42nd International Symposium on Robotics*. VDE Verlag GMBH, 2011.

[38] "Motion planning," Sep 2019. [Online]. Available: https://robotics.umich.edu/research/focus-areas/motion-planning/

[39] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 93–100.

[40] F. Farshidian, M. Kamgarpour, D. Pardo, and J. Buchli, "Sequential linear quadratic optimal control for nonlinear switched systems," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 1463–1469, 2017.

[41] J. Pankert and M. Hutter, "Perceptive model predictive control for continuous mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6177–6184, 2020.

[42] M. Fortin and R. Glowinski, *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*. Elsevier, 2000.

[43] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1366–1373.